Projektiranje digitalnog sklopa potpunog zbrajala

Hrska, Željko

VELEUČILIŠTE U KARLOVCU
STROJARSKI ODJEL
Stručni studij Mehatronike

Željko Hrska

Projektiranje digitalnog sklopa potpunog zbrajala

Završni rad

Karlovac, 2017. godina

VELEUČILIŠTE U KARLOVCU
STROJARSKI ODJEL
Stručni studij Mehatronike

Željko Hrska

Projektiranje digitalnog sklopa potpunog zbrajala

Završni rad

mr. sc. Vedran Vyroubal

Karlovac, 2017. godina

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se dragom Bogu što mi je poklonio ovaj život na ovoj prekrasnoj Zemlji, što me uvijek vodio kroz život i što je uvijek uz mene. Nadalje se zahvaljujem dragom gospodinu Bruni Gröningu na svakoj podršci, vodstvu i savjetu, koji sam smio dobiti, te također na pomoći. Zahvaljujem se gospodinu Braci i na pomoći koju sam putem njega smio primiti. Zahvaljujem se svojoj obitelji na podršci te što su me svih ovih godina trpili i imali me rado u obitelji. Nadalje, zahvaljujem se učiteljima i profesorima koji su me vodili tijekom školovanja i potaknuli da postignem svoje ciljeve. Zahvaljujem se kolegama koji su mi pomagali tijekom studija te isto tako prijateljima što su me podržavali i pomogli savjetima. Zahvaljujem se svima onima koje nisam spomenuo, a pridonijeli su mojem uspjehu. Isto tako zahvaljujem se svim znastvenicima koji su pridonijelu ovom znastvenom području o kojem smijem pisati.

Zahvaljujem se svome mentoru mr. sc. Vedranu Vyroubalu na podršci, savjetima i strpljenju koji su me vodili pri pisanju i izradi završnog rada.

Željko Hrska

VELEUČILIŠTE U KARLOVCU
KARLOVAC UNIVERSITY OF APPLIED SCIENCES
Trg J.J.Strossmayera 9
HR-47000, Karlovac, Croatia
Tel. +385 - (0)47 - 843 - 510
Fax. +385 - (0)47 - 843 – 579

HRN EN ISO 9001:2009

VELEUČILIŠTE U KARLOVCU

Stručni studij: Mehatronike

Usmjerenje: Karlovac, 3. listopada 2016.

ZADATAK ZAVRŠNOG RADA

Student: Željko Hrska Matični broj: 0112613084

Naslov: **Projektiranje digitalnog sklopa potpunog zbrajala**

Opis zadatka:

Zadatak završnog rada jest izraditi te opisati potpuno funkcionalni digitalni sklop potpunog zbrajala te rad sklopa potpunog zbrajala demonstrirati s fizičkom implementacijom. U radu će biti prikazani svi dijelovi, njihove uloge te kako je što povezano u cijelom sustavu.

U Završnom radu treba opisati tehničke i fizikalne zahtjeve te konstrukciju digitalnog sklopa. Rad treba obuhvatiti sljedeće cjeline:

1. Općeniti prikaz rada digitalnog sklopa potpunog zbarajala

2. Elektronički zahtjevi
   Uloga elektroničkih komponenti kompletnog sustava

3. Primjene uređaja
   Opis načina primjene zadanog sklopa

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
3. listopada 2016.	rujan 2017.	Petnaest dana nakon predaje rada

Mentor:	Predsjednik Ispitnog povjerenstva:
mr.sc. Vedran Vyroubal	dr.sc. Vladimir Tudić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x		Prirodni broj
y		Prirodni broj
z		Prirodni broj
U_{CC}	V	Napon napajanja
f		Izlaz logičkog sklopa
Y		Izlaz logičkog sklopa
$U_{OH\ min}$	V	Minimalan potreban napon za logičku jedinicu na ulazu
$U_{OL\ max}$	V	Maksimalan napon za logičku nulu na ulazu
$U_{IH\ min}$	V	Minimalan potreban napon logičke jedinice na izlazu
$U_{IL\ max}$	V	Minimalan napon logičke nule na izlazu
A, B		Ulazne varijable logičkog sklopa
U_{CE}	V	Pad napona između kolektora i emitera
R	Ω	Otpornik
R_a , R_b	Ω	Otpornik
T_1, T_2		Oznaka tranzistora
D_1 , D_2		Oznaka diode
U_D	V	Pad napona na diodi
R_A , R_B	Ω	Otpornik
n , i		Prirodan broj
t_z	ms	Ukupno kašnjenje signala potpunog zbrajala
t_c	ms	Vrijeme potrebno da signal prođe kroz oba EX-ILI, I, ILI vrata
t_s	ms	Vrijeme potrebno signalu da prođe kroz oba EX-ILI vrata

SAŽETAK

Računala ili kalkulator operaciju zbrajanja izvrše u jednoj sekundi, dok je čovjeku potrebno malo više vremena. U svom završnom radu digitalni sklop potpuno zbrajalo, koje je integriran u svaki kalkulator i računalo, prikazat ću njegovu logičku shemu, princip rada i demonstrirati način rada i time u jednom segmentu objasniti kako današnja računala zbrajaju. U svojem ću završnom radu prikazati fundamentalna znanja za izradu digitalnog sklopa potpunog zbrajala te njegovu fizičku implementaciju.

Ključne riječi: binarni sustav, logička algebra, logička operacija, logički sklopovi, potpuno zbrajalo.

SUMMARY

Computers or calculators finish adding numbers in a second while people need more time. This work will show how the digital circuit full-adder, which is integrated in every calculator and computer, works and its logical scheme. We will also demonstrate how the full-adder works and by doing so we will try to explain how modern computers calculate. In this work we will present the fundamental knowledge of logical systems, how to design the full-adder as well as its physical implementation.

Key words: binary system, digital logic, full-adder, logic circuits, logic operation.

1. UVOD

Digitalni su sustavi u današnje vrijeme sveprisutni, od stolnog računala, pametnih mobitela pa sve do običnog kalkulatora. Digitalne sustave također susrećemo svugdje gdje je potrebno upravljanje procesa putem računala, a najviše ih susrećemo pri obradi podataka. Njihova je zadaća obrada podataka, obavljanje aritmetičkih i logičkih operacija te donošenje odluka za koje su programirani ili projektirani.

Već u školi djeca se susreću s kalkulatorom i uče kako obavljati matematičke operacije pomoću njega. On izvrši zbrajanje velikih brojeva u jednoj sekundi. Svaki digitalni kalkulator ili računalo u sebi ima integrirano potpuno zbrajalo koje, kako sam naziv kaže, zbraja, no kako uopće digitalni sustavi vide brojeve?

Jedan od ciljeva završnog rada jest u prvom poglavlju prikazati razvoj strojeva za izvršavanje raznih matematičkih operacija, kao i čovjekove potrebe za takvim strojevima i njihov povijesni razvoj. Drugo se poglavlje bavi osnovnim principima logičke algebre, tzv. Booleove algebre koja se upotrebljava u sintezi i analizi logičkih sklopova, tj. u projektiranju logičkih sklopova. Nadalje, u trećem ću poglavlju prikazati i opisati osnovne principe rada potpunog zbrajala i time zaokružiti osnovno znanje koje je potrebno za projektiranje digitalnog sklopa potpunog zbrajala. Potpuno zbrajalo izuzetno je važan logički sklop digitalnih računala zbog činjenice da se sve matematičke operacije u konačnici svode na zbrajanje binarnih brojeva. U sljedećem poglavlju biti će prikazano elektroničke komponente za projektiranja digitalnog sklopa potpunog zbrajala, od logičke sheme pa sve do fizičke implementacije pomoću elektroničkih komponenata (otpornika, tranzistora, itd.).

Cilj završnog rada jest predstaviti stečeno znanje tijekom studija te prikazati implementaciju potpunog zbrajala s dva 4-bitna broja i demonstraciju rada takvog digitalnog sklopa.

2. Povijest digitalne tehnologije

Aristotel je pojam tehnologije uveo davne 330. godine prije Krista, tj. podijelio je znanstvene spoznajna na tri dijela:

- teoretske znanosti
- praktičke znanosti
- tehnologija.

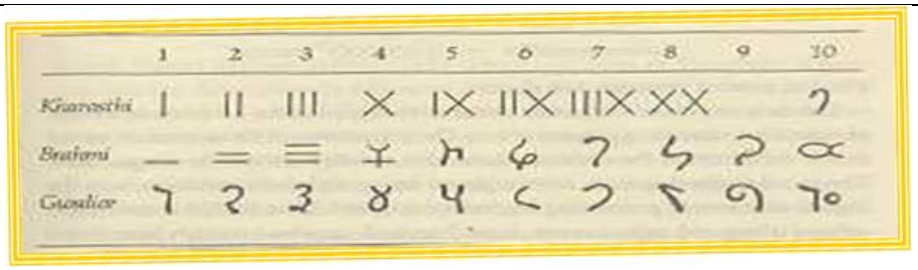
Od tog vremena sve do dan danas se vodi rasprava o tome što je tehnologija. Je li to otkriće vatre, kotača ili nečeg trećeg? Bez obzira na to što je točno tehnologija, ona je oduvijek imala veliki utjecaj na ljude i na društvo, jednom riječju, na sveukupnu civilizaciju. Dan danas čujemo kako se lako barata tom riječju "tehnologija". Kada je nešto novo, bolje rečeno napredno, često govorimo o novoj tehnologiji i o napretku, dok za neke "starije" stvari govorimo o staroj tehnologiji ili o nečemu što je već zastarjelo. Kako god bilo, pojam „tehnologija" često vežemo za znanost i napredak civilizacije. No, kako je digitalno zbrajalo pridonijelo razvitku civilizacije?

Riječ civilizacija latinskog je podrijetla i označava skup svih znanja, vještina, običaja i duhovnih spoznaja razvijene ljudske zajednice. Kada pogledamo malo u povijest, često velike zajednice ljudi koje su razvile svoje pismo i način zapisivanja nazivamo drevne civilizacije. Uz razvoj pisma paralelno se odvijao i razvoj brojeva, time i same matematike. Matematika se razvila iz potrebe za proračunima u trgovini, mjerenju zemljišta i predviđanju astronomskih događaja i pojava.

U daljnjem tekstu prezentirat ću mali pregled razvoja današnjeg brojevnog sustava i najvažnijih civilizacija koje su utjecale na zapadni svijet.

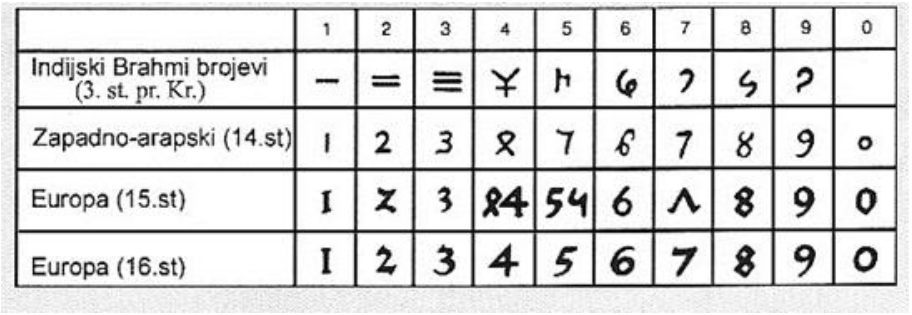
Današnji brojevni sustav, tj. brojevi potječu iz drevnog arapskog svijeta. Arapi su zapravo preuzeli brojevni sustav od drevne Indije. U drevnoj su se Indiji razvila tri brojevna sustava, a svima je zajedničko što počivaju na bazi 10 (Slika 1.):

- KHAROSTHI (oko 400 - 200 pr. Kr.)
- BRAHMI (oko 300 pr. Kr.)
- GWALIOR (oko 850 pr. Kr.)



Slika 1. Drevni indijski brojevni sustav

Muhammad Idu Musa-Al-Khwarizmi (svećenik i matematičar koji je živio u Bagdadu) uvodi indijski pozicijski brojevni sustav i u arapski brojevni sustav uvodi nulu.



Slika 2. Razvoj zapisa dekatskih brojeva

U današnje vrijeme koristi se i brojevni sustav starih Rimljana. Rimski brojevni sustav je aditivno-supraktivni sustav koji se temelji na 7 simbola koji označuju određene brojeve, pribrajanjem ili oduzimanjem jednog broja od drugoga dobije se željeni broj.

I	1	XXI	21	XLI	41	LXI	61	LXXXI	81
II	2	XXII	22	XLII	42	LXII	62	LXXXII	82
III	3	XXIII	23	XLIII	43	LXIII	63	LXXXIII	83
IV	4	XXIV	24	XLIV	44	LXIV	64	LXXXIV	84
V	5	XXV	25	XLV	45	LXV	65	LXXXV	85
VI	6	XXVI	26	XLVI	46	LXVI	66	LXXXVI	86
VII	7	XXVII	27	XLVII	47	LXVII	67	LXXXVII	87
VIII	8	XXVIII	28	XLVIII	48	LXVIII	68	LXXXVIII	88
IX	9	XXIX	29	XLIX	49	LXIX	69	LXXXIX	89
X	10	XXX	30	L	50	LXX	70	XC	90
XI	11	XXXI	31	LI	51	LXXI	71	XCI	91
XII	12	XXXII	32	LII	52	LXXII	72	XCII	92
XIII	13	XXXIII	33	LIII	53	LXXIII	73	XCIII	93
XIV	14	XXXIV	34	LIV	54	LXXIV	74	XCIV	94
XV	15	XXXV	35	LV	55	LXXV	75	XCV	95
XVI	16	XXXVI	36	LVI	56	LXXVI	76	XCVI	96
XVII	17	XXXVII	37	LVII	57	LXXVII	77	XCVII	97
XVIII	18	XXXVIII	38	LVIII	58	LXXVIII	78	XCVIII	98
XIX	19	XXXIX	39	LIX	59	LXXIX	79	XCIX	99
XX	20	XL	40	LX	60	LXXX	80	C	100
								D	500
								M	1000

Slika 3. Rimski brojevni sustav

Primjer 1:

1974. godina – MCMLXXIV

Tu vidimo da je M = 1000 (veći ispred manjega, aditivno svojstvo), sad je manji ispred većeg M (1000) - C (100) = 900 (tu vidimo subtraktivno svojstvo), veći ispred manjih

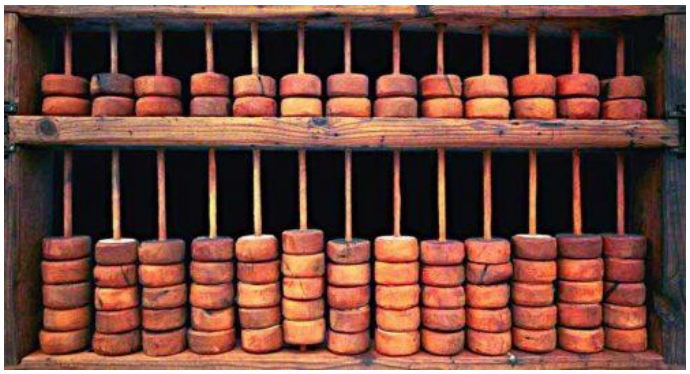
L (50) + X (10) + X (10) =70, (manji ispred većeg) I (1) - V (5) = 4.

Jedna mala zanimljivost jest što su australski starosjedioci Gumulgala imali binarni brojevni sustav temeljen na dvije riječi urapon i ukasar, i to prije pojave binarnog sustava.

Primjer 2:

broj 5 je ukasar - ukasar - urapon (2+2+1)

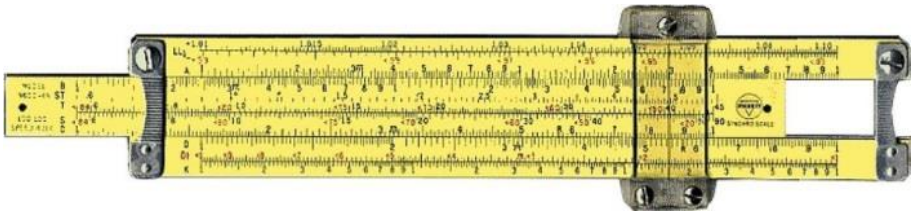
Kolika je bila potreba za matematikom, bolje rečeno za brzim zbrajanjem i oduzimanjem, svjedoči nam abakus. Abacus (Abak) jedna je od prvih naprava za računanje, tj. za zbrajanje i oduzimanje. Potječe iz stare Kine i konstruiran je oko 3000 g.pr.Kr. Koristili su ga stari Grci i Rimljani. U Europi se koristio sve do 17. stoljeća, do pojave arapskih brojeva i računanja po papiru.



Slika 4. Abacus

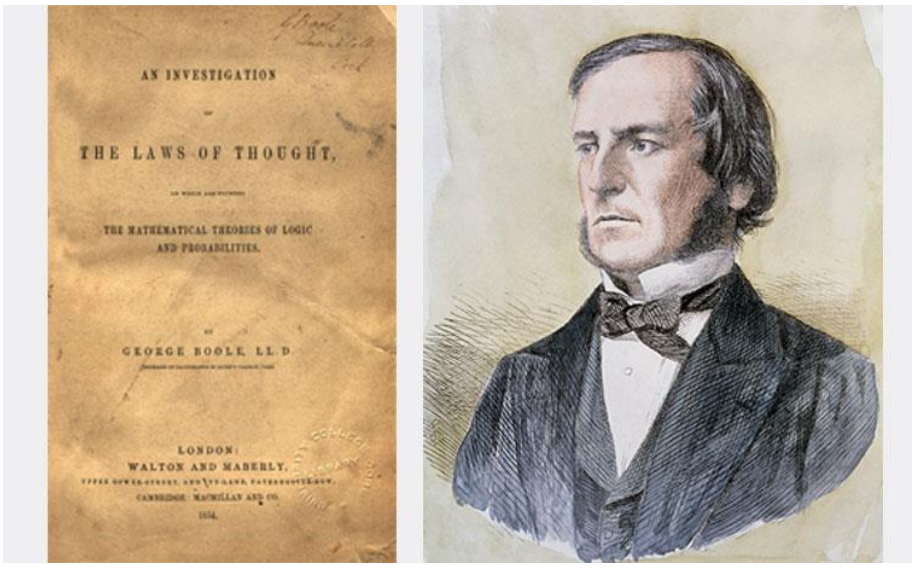
Kako se razvijala matematika, tako se javljala potreba za što bržim računanjem. Leonardo da Vinci u 15. stoljeću izradio je nacrt za prvi stroj za računanje CODEX, ali on nikada nije realiziran. Tek je u 20. stoljeću konstruiran. U 17. stoljeću škotski matematičar John Napier uočio je da se računske operacije množenja i dijeljenja mogu svesti na zbrajanje i oduzimanje. Poznat je zato što je prvi objavio logaritamske tablice. Godine 1617. izradioje računaljku sličnu abaku (tzv. „Napierove kosti“). Kasnije, 1622. godine engleski matematičar William Oughtred konstruirao je logaritmar koji po svojoj konstrukciji podsjeća na ravnalo koje klizi središtem drugog ravnala (tzv. Logaritamski šiber). Zbog malih dimenzija i lakoće uporabe, logaritmar se koristio do nedavno, poglavito u tehničkim strukama. Prvi mehanički kalkulator izumio je Wilhelm Schickard 1623. godine, koji je Johannes Kepler koristio za svoje

proračune. Kalkulator nije sačuvan jer je uništen u požaru, ali je u 19. stoljeću napravljena njegova replika. Blaise Pascal patentirao je 1642. godine prvi mehanički stroj s mnoštvom zupčanika koji je zbrajao i oduzimao velike brojeve. Nazvao ga je Pascalina. Bio je neprecizan zbog tadašnje tehnologije izrade, štoviše izvodio je samo operacije za koje je napravljen.



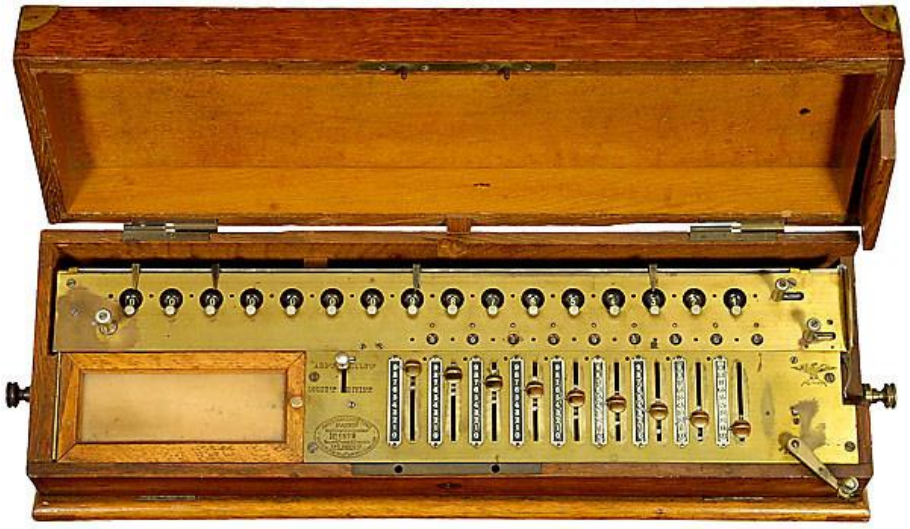
Slika 5. Logaritmar

Njemački filozof i matematičar Gottfried Wilhelm Leibniz izradio je 1672. godine mehanički kalkulator koji je izvodio sve četiri osnovne matematičke operacije (zbrajanje, oduzimanje, množenje i dijeljenje). Leibniz utemeljio je 1705. godine binarni brojevni sustav kakav danas poznajemo te zaključio kako je binarni brojevni sustav najpogodniji za strojeve koji se bave računanjem. Isto tako implementacijom binarnog brojevnog sustava moguće je koristiti principe iz aritmetike i logike te ih kombinirati. Kasnije, 1894. godine George Boole izdat će svoje djelo *The Law of Thought* i time utemeljiti Booleovu algebru (digitalnu algebru) koja je osnova svih modernih računala.



Slika 6. The Law of Thought i George Bool

Francuz Charles Xavier Thomas de Colmar na temelju Pascalovog i Leibnizovog izuma konstruirao je 1820. godine prvi komercijalni mehanički kalkulator *Arithmometer* koji je bio u prodaji sve do 1935. godine.



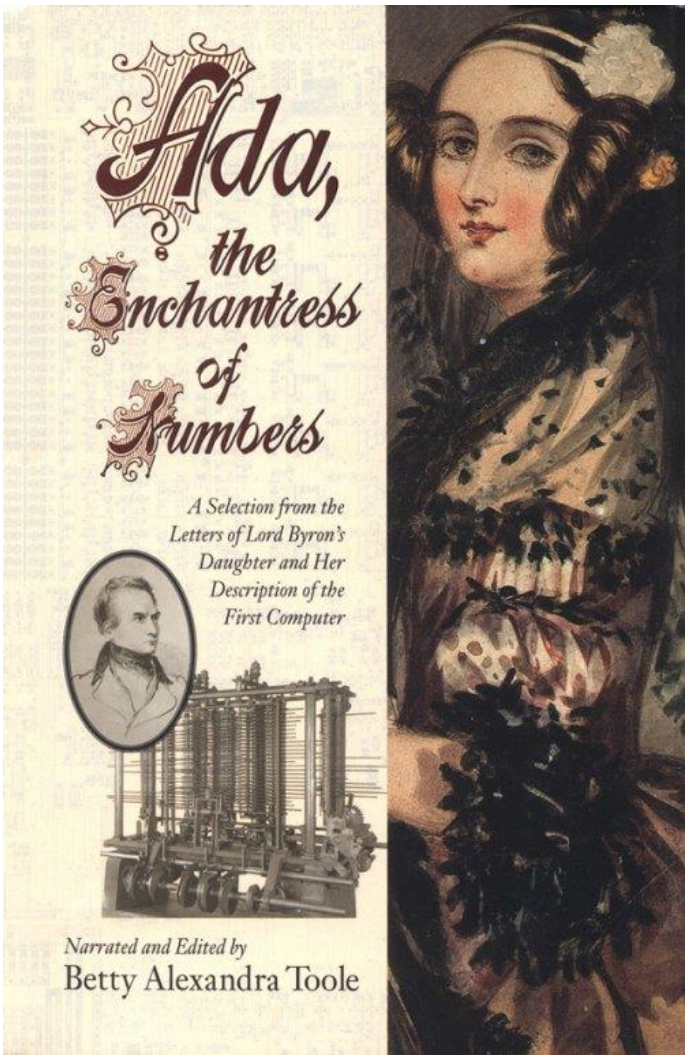
Slika 7. Arithmometer

Engleski matematičar Charles Babbage, poznat kao „otac računalstva“, uočio je nedostatak računalnih strojeva te je 1833. godine osmislio i napravio prototip Diferencijalnog stroja. Taj stroj bio je potpuno automatiziran, pokretan vodenom parom i rješavao je različite zadatke, kao i računanje logaritamskih tablica. Sljedeći Babbageov stroj bio je Analitički stroj (engl. *Analitical Engine*) preteča današnjih strojeva, no nije realiziran zbog tadašnjih tehnoloških ograničenja. Funkcije stroja podijelio je u dijelove:

- dio koji nosi podatke
- dio koji računa
- dio koji daje instrukcije
- dio koji pamti podatke
- dio koji prikazuje podatke.

Već tu vidimo koncept današnjih računala. Stroj je trebao primati naredbe, tj. programe u obliku bušenih kartica. Kontesa od Lovelace Ada Augusta Byron, matematičarka i znanstvena kolegica Charlesa Babbagea, pisala je prve programe za analitički stroj kako bi ti programi bili spremni kada se stroj napravi. Smatra se prvom programerkom i programski jezik ADA dobio je ime njoj u čast. Charles Sanders Perice ustanovio je i opisao kako se logičke operacije mogu realizirati pomoću sklopki. Zahvaljujući napretku u području elektrotehnike, 1887. godine Herman Holerith upotrebom tehnologije elektromagneta konstruirao je prvi elektromehanički stroj- Sortni stroj (engl. *Tabulating Machine*) koji je koristio bušene kartice. Stroj je mogao sortirati i jako brzo zbrajati, ali nije mogao rješavati složenije zadatke. Ovim je

strojem 1890. godine riješen problem obrade rezultata popisa stanovništva u SAD-u čija je obrada svedena s nekoliko godina na mjesec dana.

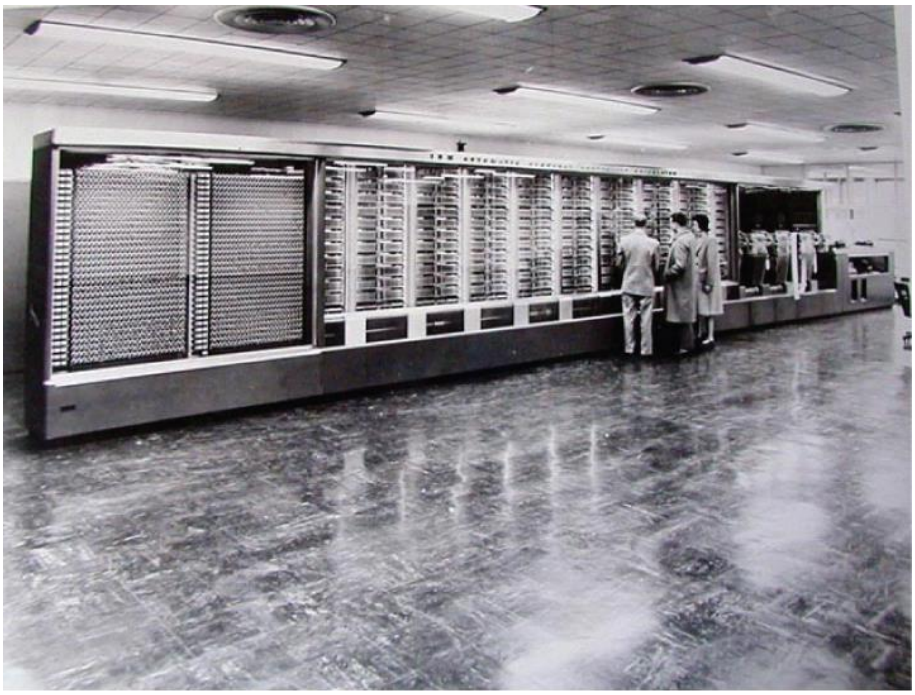


Slika 8. Charles Babbage (u krugu) i Ada Augusta Byron

Stroj se pokazao jako praktičnim i korisnim pa je Herman Holerith osnovao tvrtku *Computing Tabulating Recording Company*, koja se 1924. godine udružila s još nekoliko tvrtki i prerasla u *International Buisness Machine* (IBM), a koja je do dan danas jedna od najznačajnijih proizvođača računala na svijetu.

U doba 2. svjetskog rata potreba za obradom podataka i potreba za računanjem naglo je porasla, što je dovelo do značajnijeg razvoja elektromehaničkih računala. U Engleskoj je 1943. godine, na temelju ideja matematičara Alana Turinga, izrađen binarni elektromehanički stroj *Collossus* kojim su uspjeli dešifrirati poruke njemačke vojske. Alan Turing dao je model univerzalnog računala, tzv. Turingov stroj i time postavio matematički apstraktni model računala. On je prvi koji je računalnim strojevima dao ime *computer* od engleske riječi *to cumpute* (računati). U Njemačkoj je Conrad Zuse 1937. godine konstruirao Z1 računalo, tj.

programabilno računalo temeljeno na binarnom sustavu. Godine 1943. konstruirao je Z3, prvo relejno elektromehaničko računalo koje je imalo tastaturu za unos podataka, a izlaz je prikazan pomoću lampica. To je bilo prvo potpuno funkcionalno automatsko računalo gdje se rad računala regulirao programom. U Americi Howard Aiken vođen idejom i vizijom Charles Babbagea predlaže 1937. godine uređaj MARK 1 koji je realiziran 1944. godine. To je potpuno automatizirano elektromehaničko računalo koje je bilo 17 metara dugačko i 2,5 metara visoko, a sastojalo se od nešto manje od milijun dijelova.

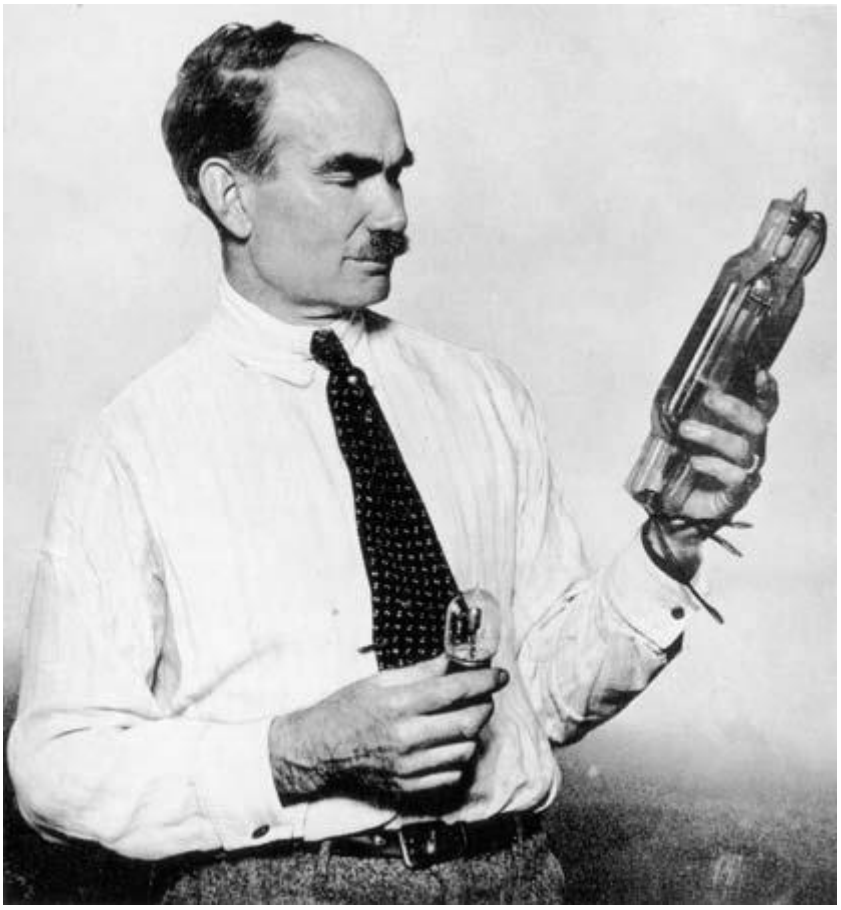


Slika 9. MARK 1

Zahvaljujući Karlu Ferdinandu Braunu, koji je 1897. godine predstavio prvu katodnu cijev, John Ambrose Fleming je 1904. godine konstruirao prvu elektronsku cijev - diodu. Često se kao godinu rađanja moderne elektronike uzima 1906. godina kada je Lee De Forest konstruirao triodu. Time su elektronske cijevi počele zamjenjivati releje. Lee De Forest je 1907. godine konstruirao „I“ vrata (AND gate) i time dokazao da se elektronske cijevi mogu koristiti u digitalnoj logici kao zamjena za releje. Walther Bothe je 1954. godine dobio Nobelovu nagradu za fiziku za prva moderna potpuno elektronska „I“ vrata iz 1924. godine.

Na sveučilištu u Pennsylvania u SAD-u 1942. godine započinje projekt izrade prvog elektroničkog digitalnog računala ENIAC (*Electronic Numerical Integrator and Computer*). Projekt je završen i predstavljen javnosti 15. veljače 1946. te se taj dan smatra početkom razdoblja elektroničkih digitalnih računala. Zanimljiva je specifikacija CPU-a, mogao je izvršiti 5000 zbrajanja, 357 množenja i 38 dijeljenja u sekundi. Površinom je zauzimao 167,3

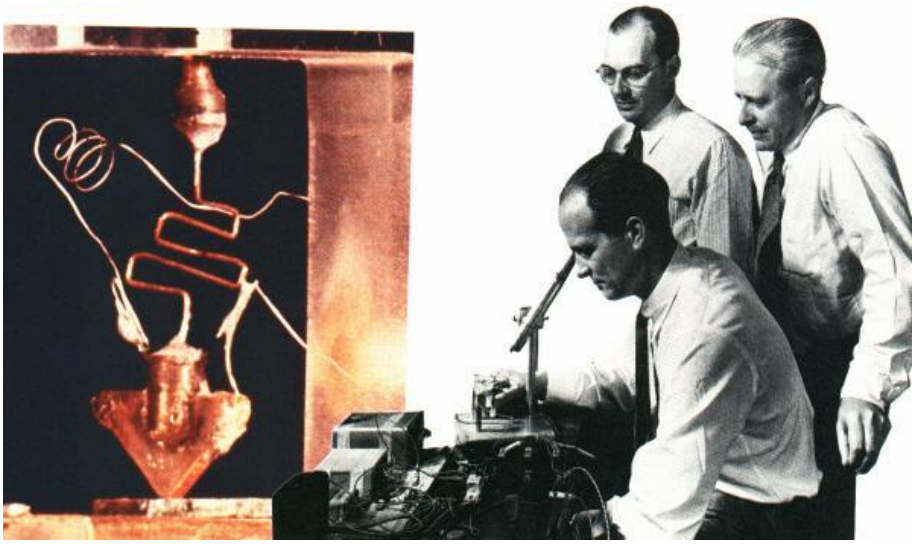
m^2. ENIAC je bio prvo računalo temeljeno na elektronskim cijevima. Uočena su dva velika nedostatka: da računalo ima malu memoriju i da može rješavati samo dva zadatka za koje je konstruiran te da je za druge zadatke trebalo rekonstruirati stroj i prespajati jako puno žica.



Slika 10. Lee de Forest i trioda (desno)

Usprkos tome ENIAC je bio za tadašnje prilike jako brzo računalo, ali nije bilo programabilno u današnjem smislu riječi. Vidjevši taj nedostatak, mađarski matematičar John von Neumann daje ideju prema kojoj bi računalo radilo na osnovi izmjenjivog programa. Još i dan danas moderna računala imaju tu arhitekturu, tzv. Von Neumannov model arhitekture računala. Prekretnica u razvoju digitalne tehnologije bio je izum tranzistora Johna Bardena i Waltera Brattaina, pod vodstvom Williama Shockleya, koji su ga u prosincu 1947. godine konstruirali u *Bell Telephone* laboratoriju. Već 1948. godine Willian Shockley unapređuje taj tranzistor i konstruira prvi bipolarni tranzistor od germanija. Prvi silicijski tranzistor proizveden je u *Texas Instruments* kompaniji 1954. godine. Pojavom tranzistora uslijedio je veoma brzi razvoj poluvodičke tehnologije. Počinje se uvoditi termin tranzistorska tehnologija. Od 1955. godine tranzistori su zamijenili elektronske cijevi u konstrukciji računala. U odnosu na elektronske cijevi tranzistor se manje grije i pouzdaniji su u radu, što je dovelo do smanjenja dimenzija računala, njihove stabilnosti i veće brzine rada te puno

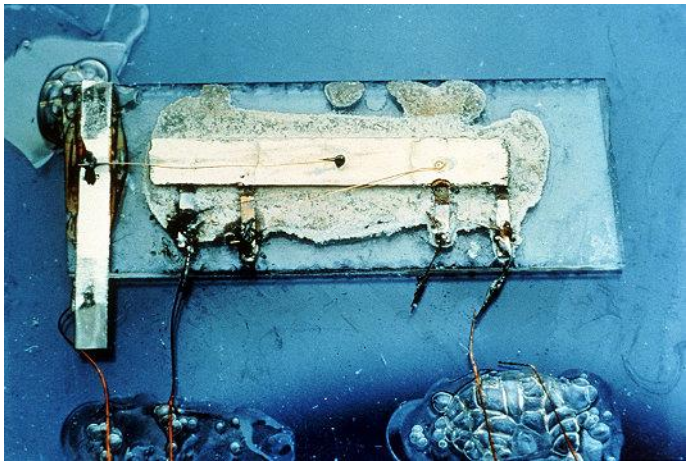
jeftinije izrade. Na Sveučilištu u Manchesteru grupa znanstvenika, koju je predvodio Tom Kilburn, dizajnirala je prvo računalo temeljeno na tranzistorskoj tehnologiji koje je predstavljeno 1953. godine. Uslijedio je brzi razvoj poluvodičke tehnologije i proizvedene su brojne vrste tranzistora. Prvi unipolarni MOS (*Metal Oxide Semiconductor*) tranzistor proizveden je 1960. godine.



Slika 11. Prvi tranzistor, (s lijeva nadesno) W. Shockley, J. Barden i W. Brattain

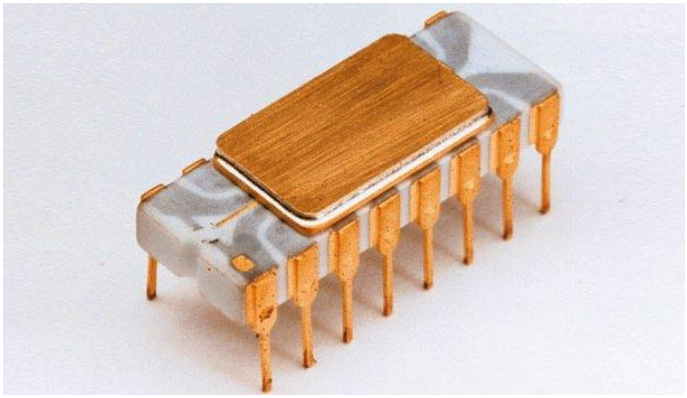
Mogućnost i težnja za minijaturizacijom i koncentracijom što većeg broja elemenata na malom prostoru dovela je do razvoja integriranih krugova. Radeći na minijaturizaciji, američki istraživač u *Texas Instruments* kompaniji Jack Kilby došao je na ideju 1958. godine da s prikladnom tehnologijom na silicijsku pločicu smjesti poluvodičke elemente i međusobno ih spoji s pasivnim elektroničkim elementima. Tako je patentirao principe integracije i napravio prvi prototip integriranog sklopa. Za to postignuće 2000. godine dobio je Nobelovu nagradu za fiziku. Paralelno je Robert Noyce iz *Fairchild Semiconductor* kompanije osmislio način kako spojiti elektroničke elemente koristeći metalizaciju aluminijem i poboljšao je izolaciju između elemenata integriranog kruga. Prvi komercijalno dostupan monolitni integrirani sklop proizvela je kompanija *Fairchild* 1961. godine. Pojavom integriranih sklopova započelo je razdoblje mikroelektronike. Suvremeni mikroelektronički sklopovi visokog stupnja integracije sadrže i nekoliko stotina tisuća elemenata povezanih u jednu cjelinu popularno zvan ČIP. Koristeći iskustva računalne tehnike iz doba elektronskih cijevi i diskretnih poluvodičkih elemenata, istraživači tvrtke INTEL Fred Faggin i Marcian Hoff izradili su 1970. godine prvi mikroprocesor (Slika 13)- pločicu silicija malih dimenzija na kojoj su smješteni svi elektronički elementi potrebni za rad središnje jedinice računala (*Central Processor Unit*, skraćeno CPU). Bio je to revolucionaran izum koji je potpuno

izmijenio daljnji tijek razvoja informatičke tehnologije. Zbog malih dimenzija, male potrošnje, a velike sposobnosti obrade podataka, pojava mikroprocesora omogućila je smanjenje dimenzija tadašnjih računala.



Slika 12. Prvi integrirani krug

Isto tako mikroprocesor je naišao na široku primjenu u raznim industrijskim granama i u svakodnevnoj uporabi. Time je otvoreno novo razdoblje sprege mikroelektronike i računalstva. Gordon Earl Moore, jedan od osnivača INTEL-a, godine 1965. predvidio je tehnološki rast novih generacija integriranih krugova. Postavio je poznati Mooreov zakon koji kaže da se svakih 18-24 mjeseci broj tranzistora na čipovima udvostručuje. Taj je zakon izdržao sve do 2001. godine kada je modificiran i glasi: „Broj tranzistora na čipovima udvostručuje se svakih 4-5 godina (48-60 mjeseci).“



Slika 13. Prvi mikroprocesor Intel 4004

U prošlom su stoljeću integrirani krugovi rađeni u 500 nanometarskom proizvodnom procesu. U današnje vrijeme kompanije rade na uporabi nanotehnologije u rješavanju kompleksnih inženjerskih problema kod proizvodnje integriranih krugova veličine 45 nm, 30 nm i manjih. U ožujku 2005. godine Craig Barrett, CEO INTEL-a, izjavio je da je tradicionalna tehnologija izrade integriranih krugova CMOS (*Complementary Metal Oxide Semiconductors*)

tehnologija. U današnje vrijeme sve je više istraživanja u smjeru tranzistora od ugljičnih nanocijevi koji bi po performansama trebali nadmašiti silicijske tranzistore i time omogućiti Mooreovom zakonu miran život.

Tablica 1. Prikaz razvoja čipova

SKRAĆENO	**PUNI NAZIV**	**GODINA**	**BROJ TRANZISTORA**	**BROJ LOGIČKIH VRATA**
SSI	*Small-scale integration*	1964.	1 - 10	1 - 12
MSI	*Medium-scale integration*	1968.	10 - 500	13 - 99
LSI	*Large-scale integration*	1971.	500 - 20000	100 - 9999
VLSI	*Very large-scale integration*	1980.	20000 - 1000000	10000 - 99999
ULSI	*Ultra large-scale intergration*	1984.	1000000 - na više	100000 - na više

3. Logička algebra

Za logičku algebru često se koristi sinonim Booleova algebra, koja je sustav teorema koji rabe simboličku logiku kako bi opisali skupove elemenata i odnose među njima. Booleova algebra dobila je ime po svom tvorcu George Booleu koji je bio poznati engleski matematičar, učitelj i filozof. Rođen je 2. studenog 1815. u Lincoldu, Engleska. U svojoj 19. godini uspio je osnovati svoju privatnu školu u Lincoldu. Od 1838. godine priznat je od strane Britanske akademske zajednice te je još više radio na područjima matematike i logike. Svoje kapitalno djelo *The Laws of Thought* (matematička analiza logike) izdao je 1854. godine, a među poglavljima tog djela nalazi se i logička algebra. U svom djelu želio je matematički obraditi postupke logičkog zaključivanja, pri čemu ulazni podaci mogu imati jedno od dvaju stanja: istinito ili lažno. Umro je 8. prosinca 1864. Njemu u čast logička algebra zove se i Booleova algebra. Osnovni element logičke algebre jest sud. To je izjavna rečenica koja može biti istinita ili lažna, ali ne oboje u isto vrijeme. Temeljno svojstvo suda jest istinitost ili lažnost. Istinitost suda označavamo simbolima T (*true*) ili 1 ili P, a lažnost suda simbolima F (*false*) ili 0 ili A. Sudove (operande) možemo kombinirati u logičke izraze koje povezujemo vezama (operatorima). Takav sustav, u kojem je prisutno samo dva stanja, isti je kao i binarni sustav brojeva u kojem su samo dva broja ili stanja: 0 ili 1. Gottfried Wilhelm Leibniz ustanovio je da je binarni sustav najbolji za korištenje u konstrukciji zbrajala, tj. tadašnjih kalkulatora. Primijetio je da se korištenjem binarnog brojevnog sustava mogu koristiti principi iz aritmetike i logike te ih međusobno kombinirati. Razvojem digitalnih računala ustanovljeno je kako je Booleova algebra vrlo dobro primjenjiva u konstruiranju i analizi rada računala, jer digitalna računala također imaju samo dva stanja: uključeno - isključeno (relejna logika) ili napon maksimalan - napon minimalan.

3.1. Binarni brojevni sustav

Binarni brojevni sustav način je zapisivanja brojeva i njihovog tumačenja. U uporabi su položajni brojevni sustavi. To su sustavi gdje položaji znamenki u zapisu određuje njezinu vrijednost. Svaki brojevni sustav određen je vlastitim skupom znamenki, a ukupan broj različitih znamenki naziva se bazom brojevnog sustava. Baza brojevnog sustava obično se zapisuje kao indeks nakon samog broja, a ako znamo o kojem se brojevnom sustavu radi, često zapisujemo bez indeksa. Brojevni sustav koji svakodnevno rabimo jest dekadski

brojevni sustav. Osnova tog sustava je broj 10, a to znači da brojevni sustav ima deset znamenki. Za zapis dekadskog brojevnog sustava koriste se znamenke 0,1,2,3,4,5,6,7,8,9. Sveukupno 10 znamenki. U dekadskom brojevnom sustavu svaki se cijeli broj sastoji od određenog broja jedinica, desetica, stotica, tisućica itd., a te vrijednosti nazivamo dekadskim jedinicama.

U svakom brojevnom sustavu vrijedi da svaka znamenka u nizu ima jedinstvenu težinsku vrijednost. Težinska se vrijednost svake znamenke dobije na način da se osnova brojevnog sustava potencira eksponentom tog sustava, čija vrijednost ovisi o položaju znamenke u zapisu. Dekadske se jedinice nazivaju potencijama broja 10 jer je to baza brojevnog sustava.

Zamislimo broj xyz, gdje su x, y i z znamenke dekadskog sustava. Njihov bi matematički zapis glasio:

- za jedinice je 10^0, $10^0 \cdot z$
- za desetice je 10^1, $10^1 \cdot y$
- za stotice je 10^2, $10^2 \cdot x$.

Vrijednost broja bi bila $10^0 \cdot z + 10^1 \cdot y + 10^2 \cdot x$.

Uzmimo za primjer broj 279, broj dvjestosedamdesetdevet. Njegov zapis pomoću znamenki je 279, vrijednost broja je prikazan jednadžbom (1).

$$10^2 \cdot 2 + 10^1 \cdot 7 + 10^0 \cdot 9 = 100 \cdot 2 + 10 \cdot 7 + 1 \cdot 9 = 279 \quad \textbf{(1)}$$

Broj „dvjestosedamdesetdevet“ njegov dekadski zapis je 279, a njegova dekadska vrijednost je 279.

Binarni sustav predstavlja pozicijski brojevni sustav s bazom 2, a zapisan je binarnim znamenkama 0 i 1. Broj se prikazuje nizom binarnih znamenki te ako postoji decimalni dio broja i decimalnom točkom. Svako ljevije mjesto u zapisu je za duplo veće od desnog. Prema tome, matematički zapis binarnog brojevnog sustava jest:

2^0 - najmanja vrijednost

2^1 - sljedeće mjesto u zapisu koje se nalazi lijevo u odnosu na najmanju vrijednost

2^2 - sljedeće mjesto u zapisu koje se nalazi lijevo u odnosu na 2^1

2^3 - sljedeće mjesto u zapisu koje se nalazi lijevo u odnosu na 2^2

itd.

Prema tome u usporedbi s dekadskim sustavom, težine cjelobrojnih mjesta u binarnim brojevnom sustavu su $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$ ...

Može se izgraditi binarni sustav prema Tablica 2. (do broja 16).

Tablica 2. Zapis binarnih, dekadkih i heksadecimalnih brojeva

Binarni	Dekatski	Heksadecimalni
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10

Za znamenke binarnog brojevnog sustava vrlo se često koristi naziv bit (skraćenica od engleskog izraza *binary digit*). Iz tablice možemo uočiti da se s jednom binarnom znamenkom, odnosno sa 1 bitom mogu dobiti dvije različite kombinacije, sa 2 bita četiri različite kombinacije, sa 3 bita osam različitih kombinacija itd. Danas se uglavnom koristi 8-bitni način zapisivanja, tj. može se sa 8 znamenki ostvariti 256 različitih kombinacija. Još se koristi i 16-bitni zapis, gdje je moguće 512 različitih kombinacija. Binarni se brojevi mogu pretvoriti u dekadske brojeve, a to se radi tako da se svaka znamenka binarnog broja pomnoži sa svojim težinskim mjestom (potencijom) pa se tako dobiveni iznosi zbroje.

Primjer 3.:

$$\begin{aligned} 1001101_{(2)} &= 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot 64 + 0 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= 77_{(10)} \end{aligned} \quad \textbf{(2)}$$

Jedan od načina pretvorbe decimalnog broja u binarni jest da se decimalni broj dijeli s brojem 2. Ako se pri dijeljenju dobije ostatak, tada je znamenka binarnog broja 1, a ako nema ostatka, tada je znamenka 0. Prvo dijeljenje daje znamenku nižeg brojnog mjesta.

Primjer 4.:

broj 67

67:2=33 ostatak 1

33:2=16 ostatak 1

16:2= 8 ostatak 0

8:2= 4 ostatak 0

4:2= 2 ostatak 0

2:2= 1 ostatak 0

1:2= 0 ostatak 1 **(3)**

Dobili smo dekadski broj 67 zapisan u binarnom brojevnom sustavu: $1\ 0\ 0\ 0\ 0\ 1\ 1_{(2)}$.

Heksadecimalni brojevni sustav je brojevni sustav s bazom 16, odnosno sustav ima 16 znamenaka. Za znamenke 0-9 koriste se znamenke decimalnog brojevnog sustava, a za znamenke 10-15 ne postoje simboli pa se koristi prvih šest slova abecede: A, B, C, D, E, F (prikazano Tablica 2.). Heksadecimalni brojevni sustav koristi se vrlo često u prikazivanju rada digitalnih uređaja zbog jednostavne pretvorbe u binarni brojevni sustav i obrnuto.

Primjer 5.:

Zapis broja $2B3C_{(16)}$ i pretvorba u dekadski brojevni sustav:

$$2B3C_{(16)} = 2 \cdot 16^3 + B \cdot 16^2 + 3 \cdot 16^1 + C \cdot 16^0$$
$$= 2 \cdot 4096 + 11 \cdot 256 + 3 \cdot 16 + 13 \cdot 1$$
$$= 8192 + 2816 + 48 + 13$$
$$= 11069_{(10)} \quad \textbf{(4)}$$

Pretvorba decimalnog broja u heksadecimalni može se sprovesti istim postupkom kao i kod binarnog brojevnog sustava, odnosno uzastopnim dijeljenjem decimalnog broja sa 16 (osnovicom sustava). Ostatak dijeljenja označava heksadecimalne znamenke.

Primjer 6.: broj $3576_{(10)} = ?_{(16)}$

3576:16 = 223 + ostatak 8

223:16 = 13 + ostatak 15 (F)

13:16 = 0 + ostatak 13 (D) **(5)**

Rezultat dobili smo $DF8_{(16)}$.

Broj u heksadecimalnom sustavu pretvara se u binarni broj tako da se svaka znamenka heksadecimalnog brojevnog sustava nadomjesti odgovarajućim binarnim brojem, tj. četverobitnom kombinacijom.

Primjer 7.:

$7D6_{(16)} = ?_{(2)}$

7	D	6
↓	↓	↓
0111	1101	0110

$7D6_{(16)} = 0111\ 1101\ 0110_{(2)}$

Za pretvorbu binarnog broja u heksadecimalni potrebno je binarni broj razdijeliti u skupine od četiri binarne znamenke počevši od najnižeg mjesta. Svaka skupina binarnih znamenaka predočava se ekvivalentnom heksadecimalnom znamenkom. Ukoliko se u zadnjoj grupi nađu 3 ili manje znamenki, utoliko se dodaje nula s lijeve strane dok se ne dobije 4-bitna kombinacija.

3.2. Matematika u binarnom brojevnom sustavu

Pod naslovom „Matematika u binarnom sustavu“ mislimo na binarnu aritmetiku, tj. na četiri osnovne matematičke operacije (zbrajanje, oduzimanje, množenje i dijeljenje) u binarnom sustavu. Svi znamo zbrajati, oduzimati, množiti i dijeliti u dekadskom brojevnom sustavu gdje na raspolaganju imamo 10 znamenki. No, kako zbrajati, oduzimati, množiti i dijeliti u binarnom sustavu koji ima samo dvije znamenke 0 i 1?

3.2.1. Zbrajanje u binarnom brojevnom sustavu

Prvo se prisjetimo zbrajanja u dekadskom sustavu, npr. 68+235. Brojeve prvo napišemo jedan ispod drugoga tako da bude znamenka jedinice prvog broja ispod znamenke jedinice drugog broja, a zbraja se s desna u lijevo.

$$\begin{array}{r} 68 \\ +\ 235 \\ \hline 303 \end{array} \qquad \textbf{(6)}$$

Pri tome je 8+5= 13 pa 3 pišemo i 1 pustimo dalje. Ta se jedinica naziva prijenos i zbraja se sa znamenkama iz idućeg stupca. Zbrajanje u binarnom sustavu provodi se na identičan način, s time da treba imati na umu sljedeća pravila zbrajanja binarnih brojeva:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 i 1 pamtimo dalje (prijenos).

Prijenos je zgodno pisati iznad kako ga ne bismo zaboravili.

Primjer 8.:

$$\begin{array}{r} 10011 \\ +\ 11001 \\ \hline 101100 \end{array} \quad \textbf{(7)} \qquad \begin{array}{r} 1101011 \\ +\ 10110 \\ \hline 10000001 \end{array} \quad \textbf{(8)}$$

3.2.2. Oduzimanje u binarnom brojevnom sustavu

Oduzimanje u binarnom sustavu obavlja se na jednaki način kao i u dekadskom brojevnom sustavu. Promotrimo sljedeći primjer:

$$\begin{array}{r} 53 \\ -\ 27 \\ \hline 26 \end{array} \quad \textbf{(9)}$$

Primijetimo da 3 - 7 nije moguće izračunati u skupu prirodnih brojeva pa moramo posuditi jednu jedinicu iz lijevog stupca i računamo 13 - 7 = 6, ali tu jedinicu moramo oduzeti od sljedećeg stupca. Oduzimanje u binarnom sustavu može se provesti na dva načina. Pravila oduzimanja u binarnom sustavu jest:

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1 i 1 pamtimo dalje (oduzima se od idućeg).

3.2.2.1. Prvi način oduzimanja u binarnom brojevnom sustavu

Prvi je način istovjetan oduzimanju kao i u dekadskom brojnom sustavu uz predočenje tablice oduzimanja.

Primjer 9.:

$$\begin{array}{r} 110101 \\ -\ 10011 \\ \hline 100010 \end{array} \quad \textbf{(10)}$$

3.2.2.2. Drugi način oduzimanja u binarnom brojevnom sustavu

Drugi način oduzimanja se može svesti na zbrajanje pa radimo pomoću dvojnog komplementa.

Primjer 10.:

$$\begin{array}{r} 110101 \\ -\ 10011 \\ \hline ???? \end{array} \quad \textbf{(11)}$$

Koraci:

umanjitelju s lijeve strane dopišemo nule (ako je potrebno) tako da umanjenik i umanjitelj imaju jednaki broj znamenki

010011

odradimo komplement umanjitelja (umjesto 0 piše se 1, a umjesto 1 piše se 0)

010011 → 101100

komplementu pribrojimo 1

$$\begin{array}{r} 101100 \\ +\ 1 \\ \hline 101101 \end{array} \quad \textbf{(12)}$$

dobili smo dvojni komplement 101101

dobiveni broj pribrojimo umanjeniku te odbacimo krajnju lijevu jedinicu.

$$\begin{array}{r} 110101 \\ +\ 101101 \\ \hline 1100010 \end{array} \rightarrow 100010 \rightarrow \text{tražena razlika} \quad \textbf{(13)}$$

3.2.3. Množenje u binarnom brojevnom sustavu

Množenje u binarnom sustavu svodi se na zbrajanje binarnih brojeva. Provodi se na sličan način kao u dekadskom sustavu uz pravila množenja binarnih brojeva:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

Pogledajmo primjer dekadskog množenja 38 • 13:

$$\begin{array}{r} 38 \cdot 13 \\ \hline 380 \\ +\ 114 \\ \hline 494 \end{array} \qquad \textbf{(14)}$$

Primjer 11.:

$$\begin{array}{r} 1001 \cdot 110 \\ \hline 1001 \\ 1001 \\ +\ 0000 \\ \hline 110110 \end{array} \qquad \textbf{(15)}$$

3.2.4. Dijeljenje u binarnom brojevnom sustavu

Dijeljenje u binarnom sustavu slično je kao i dijeljenje koje se uči u nižim razredima osnovne škole, tj. dijeljenje je svedeno na oduzimanje.

Primjer 12.:

$$\begin{array}{r} 910 : 14 = 65 \\ -\ 78 \\ \hline 130 \\ -\ 130 \\ \hline 0 \end{array} \qquad \textbf{(16)}$$

497 : 3 =165 i 2 ostatak

- 3

1

19

- 18

1

17

- 15

2 ostatak **(17)**

U binarnom sustavu dijeljenje se obavlja na sličan način, tj. svodi se na množenje.

Primjer 13.:

1010001 : 1001 =1001

1010

- 1001

1001 (spuštamo o sve dok ne možemo opet oduzimati)

- 1001

0 **(18)**

10001 : 11 = 101 i ostatak 10

100

- 11

101

- 11

10 ostatak **(19)**

S dijeljenjem počinjemo tako da uzmemo znamenku po znamenku djeljenika sve dok ne dobijemo broj veći od djelitelja (u prvom primjeru je tako 1<1001, idemo dalje 10<1001, 101<1001, 1010>1001 pa počinjemo s brojem 1010).

3.3. Binarni kodovi

U digitalnim sklopovima i uređajima podatci se prikazuju pomoću binarnih znamenaka. Kako bi se uz brojeve mogli prikazivati znakovi i slova, koriste se kodovi. Kod je određena kombinacija binarnih znamenki koja se dodjeljuje decimalnoj znamenki, slovu ili znaku. Ako se kodiranjem želi prikazati znamenka decimalnog brojevnog sustava, potrebno je koristiti kombinacije od najmanje četiri bita. Ta četiri bita može dati $2^4 = 16$ različitih kombinacija. Za prikaz znamenaka decimalnog brojevnog sustava potrebno je svega 10 kombinacija (brojevi od 0 do 9). Brojni su načini za kodiranje decimalnih brojeva. Najčešći su BCD, exsess-3 i Grayev kod.

3.3.1. BCD kod

Za kodiranje decimalnih znamenaki u BCD kodu (engl. *Binary Coded Decimal*) koristi se prvih deset kombinacija prirodnog binarnog četverobitnog niza. To znači da se svaka decimalna znamenka prikazuje pripadnim binarnim brojem (Tablica 3.). Stoga se ovaj kod ponekad naziva i prirodni binarno-decimalni kod ili skraćeno NBCD (engl. *Natural Binary Coded Decimal*). BCD kod se naziva i težinski kod jer bitovi imaju težine mjesta 8, 4, 2, 1. Kod sadrži kombinaciju 0000, što može značiti i prekid u prijenosu podataka, a može biti shvaćen i kao podatak 0. Potrebno je razlikovati broj prikazan u binarnom brojevnom sustavu od istog broja prikazanog u binarnom kodu, budući da se u oba slučaja radi o nizu bitova. Kombinacija bitova u binarnom brojevnom sustavu uvijek označava određeni broj, dok kombinacija bitova u kodu može označavati broj, ali i znakove ili slova, općenito neki podatak.

Primjer 14.: Binarna kombinacija 10010110 kao binarni broj odgovara u decimalnom sustavu broju 150, dok ta ista kombinacija u BCD kodu odgovara broju 96.

Tablica 3. Prikaz kodiranja decimalnih brojeva u BCD kodu

Dekatska znamenka	Binarna kombinacija
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

3.3.2. Excess-3 kod

Za kodiranje decimalnih znamenaka u excess-3 kodu (XS-3 kod) koristi se srednjih deset kombinacija binarnog četverobitnog niza, a odbacuju se prve i zadnje tri kombinacije. XS-3 kod razlikuje se od BCD koda po tome što nije težinski, ali je samokomplementirajući.

To znači da se komplement bilo koje znamenke dobije zamjenom nula s jedinicama i jedinicama s nulama. Osim toga, u XS-3 kodu ne pojavljuju se kombinacije sa sve četiri nule ni sa sve četiri jedinice što može biti korisno za otkrivanje prekida u prijenosu podataka.

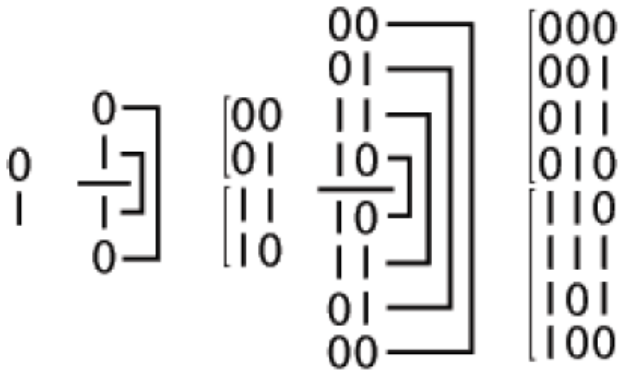
Tablica 4. Excess-3 kod

Dekatska znamenka	Binarna kombinacija
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

3.3.3. Grayev kod

Karakteristika Grayevog koda, koji nije težinski, jest da se svaka kombinacija razlikuje od prethodne za samo jedan bit. Grayev kod temelji se na reflektiranom (zrcalnom) binarnom brojevnom sustavu i zato pripada skupini reflektiranih kodova. Brojevi reflektiranog binarnog brojevnog sustava dobiju se na sljedeći način:

Znamenke 0 i 1 napišu se jedna ispod druge. Ispod njih se povuče zamišljena zrcalna crta i ispod nje se napišu znamenke 1 i 0 kao zrcalna slika. Sada se ispred gornjih znamenki dodaju nule, a ispod donjih jedinice. Na taj se način dobije skupina od četiri dvobitne kombinacije. Ako se ispod njih povuče nova zrcalna crta i ispod nje napišu reflektirane kombinacije pa se u gornjima dodaju nule, a u donjima jedinice, dobije se osam trobitnih kombinacija. Na taj se način može dobiti broj kombinacija po želji. Za Grayev kod koristi se prvih deset kombinacija četverobitnog niza. Konstruiranje Grayevog koda prikazano je na Slika 14.



Slika 14. Konstrukcija Grayevog koda

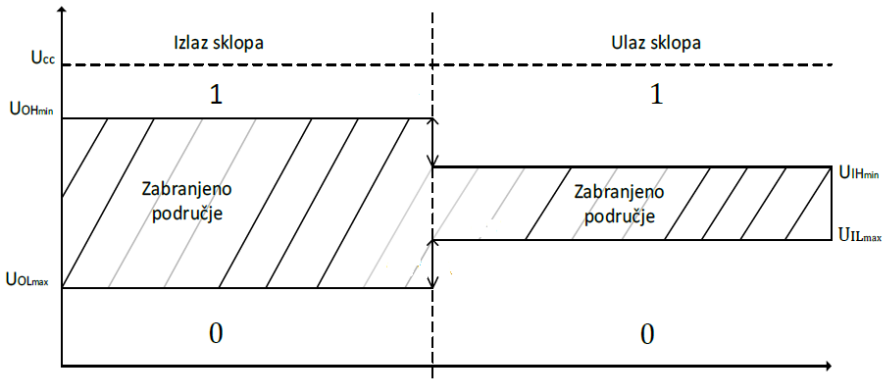
3.3.4. Usporedba kodova u zapisu dekatskih brojeva

Tablica 5. Usporedba kodova

Dekadska znamenka	BCD kod	XS-3 kod	Grayev kod
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0011
3	0011	0110	0010
4	0100	0111	0110
5	0101	1000	0111
6	0110	1001	0101
7	0111	1010	0100
8	1000	1011	1100
9	1001	1100	1101
10	0001 0000	0100 0011	1111
11	0001 0001	0100 0100	1110
12	0001 0010	0100 0101	1010
13	0001 0011	0100 0110	1011
14	0001 0100	0100 0111	1001
15	0001 0101	0100 1000	1000

3.4. Logički sklopovi

Digitalni sklopovi mogu imati jedan ili više ulaza i isto toliko izlaza. Naponi na ulazima i izlazima mogu imati vrijednosti unutar područja koje odgovaraju binarnim znamenkama 0 ili 1. Stanje napona na izlazima sklopova vezano je za ispunjenje određenih uvjeta na ulazima. Između stanja na ulazima i stanja na izlazima postoji određena logička veza, odnosno digitalni sklopovi obavljaju logičku funkciju ili operaciju. Stoga se digitalni sklopovi nazivaju i logički sklopovi. Logički sklopovi kod kojih stanje izlaza ovisi o trenutnom stanju ulaza nazivaju se kombinacijski logički sklopovi. Logička svojstva digitalnih sklopova mogu se iskazati tablicom stanja (engl. *Truth table*). Pomoću njih imamo pregledan prikaz svih kombinacija ulaznih binarnih veličina i odgovarajućih stanja na izlazu. Često se kod tvorničkih podataka proizvođača digitalnih sklopova umjesto oznaka „0“ i „1“ koriste oznake L (engl. *Low*) i H (engl: *High*), niska naponska razina i visoka naponska razina. Logički sklopovi mogu se iskazati algebarskim ili logičkim jednadžbama. U shemama digitalnih uređaja, digitalni sklopovi prikazuju se odgovarajućim simbolima. Vrlo se često koriste simboli prema američkom vojnom standardu (MIL–ST–806B, 1962. godina; *Graphic Simbols for Logic Diagrams, Department of Defense*, SAD), a od 1984. godine u uporabi je isto međunarodni standard IEC/ISO (*International Electrotehnical Commision*).



Slika 15. Prikaz razine napona i stanja digitalnih sklopova

3.4.1. Logički sklop NE

Logički sklop NE, odnosno inverter (engl. *NOT inverter, NOT gate*) obavlja logičku operaciju NE (negacija, inverzija, komplementiranje). Sklop ima jedan ulaz i jedan izlaz, što uključuje jedan operand i jedan operator. Na izlazu daje stanje suprotno stanju ulaza, tj. kad je na ulazu stanje „1“, na izlazu je stanje „0“ i obrnuto. Operator NE označava se jednim od

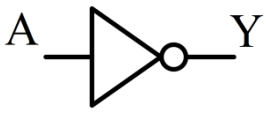
simbola: ⌐ ili ¯ ili ’. Uzmimo da je A stanje ulaza, svojstva NE sklopa su dani algebarskim izrazom, tablicom stanja i simbolom:

Algebarski izraz:

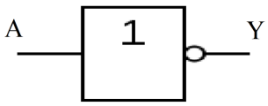
$$f = \overline{A}\,; Y = \overline{A} \tag{20}$$

Tablica 6. Tablica stanja NE sklopa

A	Y
0	1
1	0

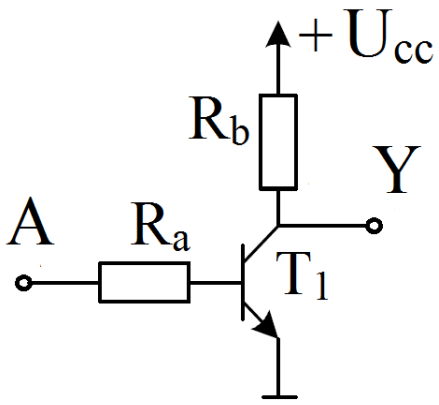


Slika 16. Simbol logičkog sklopa NE prema američkom standardu



Slika 17. Simbol logičkog sklopa NE prema IEC standardu

Fizička se implementacija radi pomoću tranzistorske sklopke (Slika 18.). Kad je na ulazu sklopa 0 V (logičko stanje „0“), radna točka tranzistora u području je zapiranja. Tranzistor se ponaša kao isključena sklopka pa je izlazni napon približno $U_{cc,}$ tj. logičko stanje „1“. Kad je na ulazu tranzistorske sklopke napona U_{cc} (logičko stanje „1“), radna točka tranzistora je u zasićenju, tranzistor se ponaša kao uključena sklopka. Na izlazu je mali napon U_{CE} što predstavlja logičko stanje „0“.



Slika 18. Izvedba NE sklopa pomoću tranzistora

3.4.2. Logički sklop I

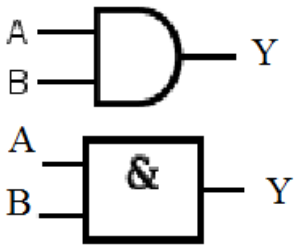
Logički sklop I (eng. AND gate) obavlja logičku operaciju I (konjunkcija ili logičko množenje). Sklop može imati dva ili više ulaza. Na izlazu daje stanje „1“, samo ako su svi ulazi u logičkoj „1“. Ako je na bilo kojem ulazu sklopa logičko stanje „0“,tada je stanje na izlazu isto „0“. Operator I označava se sa simbolima · ili & ili ∧.

Algebarski izraz:

$$f = A \cdot B \qquad Y = A \cdot B \tag{21}$$

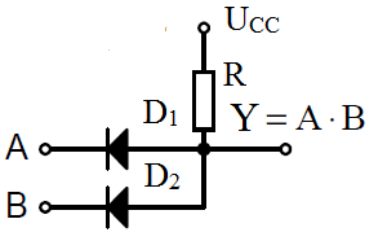
Tablica 7. Tablica stanja I sklopa

B	A	Y
0	0	0
0	1	0
1	0	0
1	1	1



Slika 19. Simbol prema američkom i IEC standardu

Sklop je moguće izvesti pomoću spoja otpornika i dioda. Ako je na oba ulaza napon 0 V (logička 0), obje će diode biti propusno polarizirane, stoga će na izlazu biti mali napon U_D (logička 0). U ovom slučaju diode predstavljaju sklopke, napon na izlazu Y ostaje U_D sve dok je barem na jednom od ulaza 0 V, jer se pripadna dioda ponaša kao uključena sklopka, tj. propusno je polarizirana. Tek kad je na svim ulazima napon U_{CC}, što odgovara logičkom stanju 1, obje diode postaju zaporno polarizirane, tj. ponašaju se kao isključene sklopke pa izlazni napon ima praktički vrijednost U_{CC}, što odgovara logičkom stanju 1.



Slika 20. Izvedba sklopa I

3.4.3. Logički sklop ILI

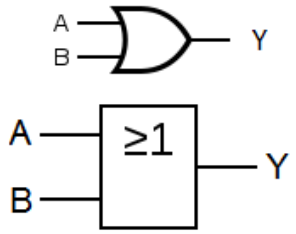
Logički sklop ILI (engl. *OR gate*) obavlja logičku operaciju ILI (inkluzivna, disjunkcija). Sklop može imati dva ili više ulaza, a na izlazu daje stanje 1 ako je na bilo kojem ulazu stanje 1. Bolje rečeno, cjelokupna logička operacija ILI je istinita ako je istinita bilo koja izjava uključena u tu operaciju. Operator ILI predočuje se jednim od simbola U ili + ili v. Najviše je u upotrebi + simbol. Sklop ovakvih svojstva je moguće izvesti spojem otpornika i dioda. Kad je na oba ulaza napon 0V, što odgovara logičkom stanju 0, obje će diode biti zaporno polarizirane, tj. ponašat će se kao isključene sklopke. Čim se barem na jednom ulazu dovede napon U_{CC}, što odgovara stanju logičke 1, dioda će se propusno polarizirati. Kroz diodu poteći će struja koja na otporniku R stvara pad napona $U_{CC} - U_D$, što odgovara logičkoj 1.

Algebarski izraz:

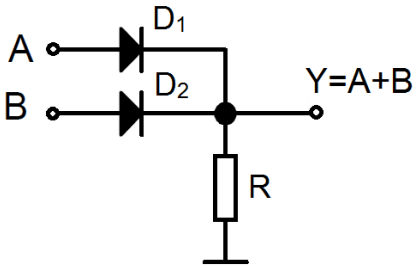
$$f = A + B = A \cup B = Y \quad \textbf{(22)}$$

Tablica 8. Tablica stanja logičkog sklopa ILI

B	A	Y
0	0	0
0	1	1
1	0	1
1	1	1



Slika 21. Američki i IEC simbol ILI vrata



Slika 22. Izvedba ILI logičkog sklopa

3.4.4. Logički sklop NI

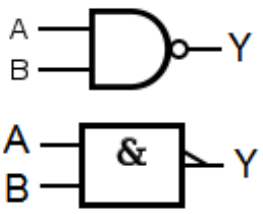
Često se u skupinu osnovnih logičkih sklopova svrstavaju izvedeni logički sklopovi. Kombinacijom osnovnih logičkih sklopova mogu se izvesti svi ostali logički sklopovi. Logički sklop NI (engl. *NAND gate*) izvedeni je logički sklop koji je sastavljen od logičkih sklopova I i NE, tj. negacija I sklopa. Logički sklop NI obavlja logičku operaciju NI (naziva se još i Shaefferova funkcija). Sklop može imati dva ili više ulaza. Na izlazu ima stanje logičke 1 ako je na bilo kojem ulazu stanje logičke 0. Kad je na svim ulazima dovedena logička 1, tada je na izlazu logička 0. Svojstva sklopa NI mogu se izvesti spajanjem diodnog sklopa I i tranzistorske sklopke NE, no sklop je moguće izvesti i kaskadnim spojem dviju tranzistorskih sklopki. Ako je na oba ulaza sklopa napon 0V, logička 1, ovakvo stanje ostaje na izlazu sve dok se na oba ulaza ne dovede napon U_{CC} (logička 1). Tada oba tranzistora prelaze u stanje zasićenja, tj. djeluju kao uključene sklopke pa se izlaz spaja s masom i na izlazu je mali napon, tj. logičko stanje 0.

Algebarski izraz:

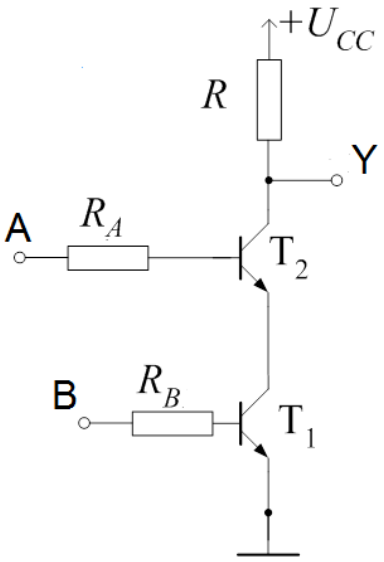
$$f = \overline{A \cdot B} = Y \tag{23}$$

Tablica 9. Tablica stanja NI sklopa

B	A	Y
0	0	1
0	1	1
1	0	1
1	1	0



Slika 23. Američki i IEC simbol NI sklopa



Slika 24. Izvedba NI sklopa pomoću tranzistora

3.4.5. Logički sklop NILI

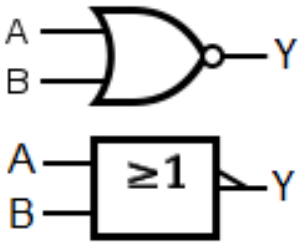
Logički sklop NILI (eng. *NOR gate*) obavlja logičku operaciju NILI (Pierceova funkcija). To je također izvedeni sklop, no u biti je negacija ILI sklopa. Sklop može imati dva ili više ulaza. Na izlazu ima stanje 1 samo kada su svi ulazi u stanju logičke 0. Kad je na bilo kojem ulazu stanje 1, na izlazu je stanje 0. Svojstva logičkog sklopa NILI mogu se izvesti spajanjem diodnog sklopa ILI i tranzistorske sklopke NE. Sklop je moguće izvesti i paralelnim spojem tranzistorskih sklopki. Ako je na oba ulaza sklopa napon 0 V, tj. logička 0, onda su radne točke tranzistora u području zapiranja i ponašaju se kao isključene sklopke pa je na izlazu Y napon U_{CC}, što odgovara logičkom stanju 1. Čim se na jedan od ulaza sklopa dovede visoki napon, tj. logička 1, pripadni tranzistor prelazi u stanje zasićenja pa se tranzistor ponaša kao uključena sklopka. Izlaz se spaja na masu i na izlazu je mali napon U_{CE}, tj. logičko stanje 0.

Algebarski izraz:

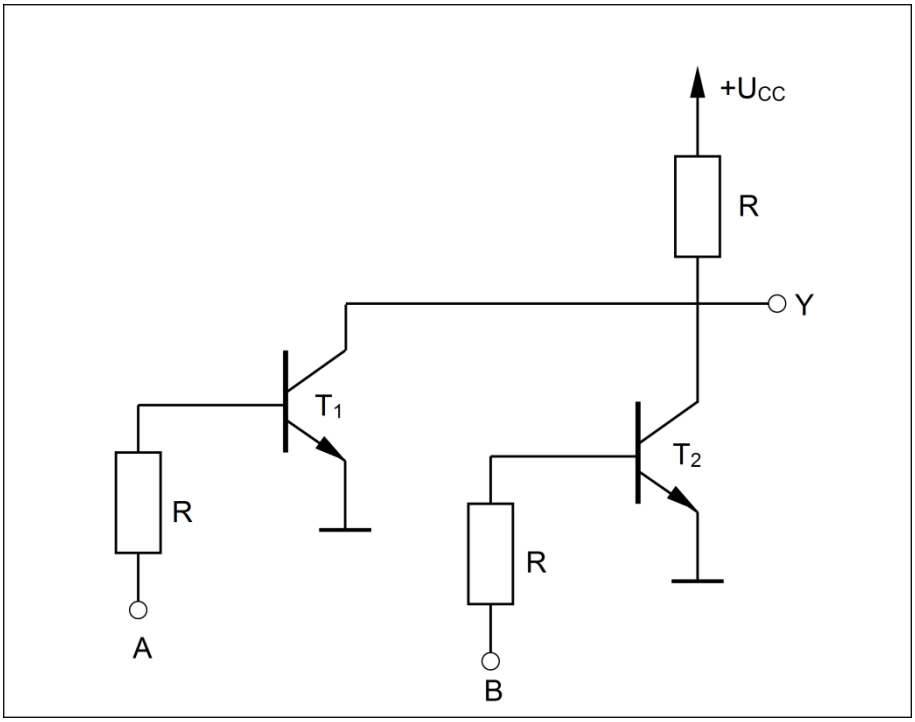
$$f = \overline{A + B} = Y \tag{24}$$

Tablica 10. Tablica stanja NILI sklopa

B	A	Y
0	0	1
0	1	0
1	0	0
1	1	0



Slika 25. Američki i ICE simbol za NILI vrata



Slika 26. Izvedba NILI vrata pomoću tranzistora

3.4.6. Logički sklop Isključivo ILI

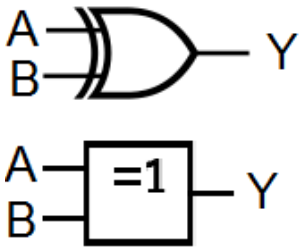
Logički sklop isključivo ILI je izvedeni sklop. Operacija isključivo ILI, skraćeno EX-ILI (engl. *Exclusive OR* ili *XOR gate*), daje na izlazu logičku jedinicu ako se ulazne varijable razlikuju, odnosno ako ima više od dviju ulaznih varijabla EX-ILI, vrata će dati logičku jedinicu samo ako postoji neparan broj logičkih jedinica na ulazu. Ako su ulazne varijable jednake, izlaz je u stanju logičke 0. Operacija isključivo ILI naziva se još i antivalencija. Isključiv ILI vrata predstavljaju i binarni komparator. Budući da u Booleovoj algebri nema istovjetnog operatera, EX-ILI prikazuje se kao $A\overline{B} + \overline{A}B$. To pomaže u Booleovoj algebri kao i zamjenjivanje izraza logičkih vrata EX-ILI.

Algebarski izraz:

$$f = A \oplus B = A\overline{B} + \overline{A}B = Y \quad \textbf{(25)}$$

Tablica 11. Tablica stanja za EX-ILI logički sklop

B	A	Y
0	0	0
0	1	1
1	0	1
1	1	0



Slika 27. Simbol EX-ILI po američkom i IEC standardu

3.4.7. Logički sklop Isključivo NILI

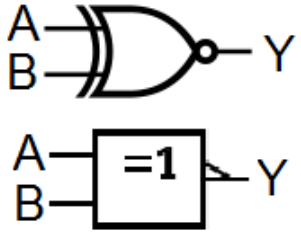
Logički sklop isključivo NILI je izvedeni sklop. Operacija isključivo NILI, skraćeno EX-NILI (engl. Exclusive NOR, XNOR gate) suprotna je operacija od isključivo ILI. To znači da će na izlazu sklopa biti u stanju logičke 1, ako su ulazne veličine međusobno jednake, odnosno ako ima više od dviju ulaznih varijabla, EX-NILI će dati na izlazu logičku 1 samo ako postoji paran broj logičkih jedinica na ulazu. Operacija EX-NILI naziva se još ekvivalencija. U Booleovoj algebri identično EX-ILI –ju, EX-NILI se prikazuje kao: $\boldsymbol{A} \cdot \boldsymbol{B} + \overline{\boldsymbol{A}} \cdot \overline{\boldsymbol{B}}$.

Algebarski izraz:

$$f = \overline{A \oplus B} = A \otimes B = A \cdot B + \overline{A} \cdot \overline{B} = Y \quad \textbf{(26)}$$

Tablica 12. Tablica stanja EX-NILI sklopa

B	A	Y
0	0	1
0	1	0
1	0	0
1	1	1



Slika 28. Simbol EX-NILI vrata po američkom i IEC standardu

4. Zakoni i teoremi Booleove algebre

Složene logičke izraze pojednostavljujemo primjenjujući teoreme (pravila) Booleove algebre. Time smanjujemo broj logičkih operacija, ali pritom ne mijenjamo tablicu istinitosti tog izraza. Rezultat primjene pravila jest ekvivalentna logička formula. Pravila za operaciju s više promjenjivih ulaznih veličina nazivaju se zakoni logičke algebre. Ti zakoni su:

I. Zakon komutacije

II. Zakon asocijacije

III. Zakon distribucije.

Među najvažnijim teoremima logičke algebre su De Morganovi teoremi. Upotrebljavaju se vrlo često u postupku pojednostavljanja logičkih operacija. Pravila za operacije s jednom promjenjivom ulaznom veličinom nazivaju se temeljna pravila logičke algebre.

4.1. Temeljni zakoni Booleove algebre

Pravila za operacije s jednom promjenjivom ulaznom varijablom prikazana su u Tablica 13., Tablica 14, Tablica 15. A je ulazna veličina koja može biti 0 ili 1.

Tablica 13. Pravila s I operatorom

$A \cdot 1 = A$	neutralni element
$A \cdot A = A$	jednaka važnost
$A \cdot \bar{A} = 0$	komplementarnost
$A \cdot 0 = 0$	anihilacija

Tablica 14. Pravila s ILI operatorom

$A + 1 = 1$	anihilacija
$A + A = A$	jednaka važnost
$A + \bar{A} = 1$	komplementarnost
$A + 0 = A$	neutralni element

Tablica 15. Involutivnost

$\overline{\bar{A}} = A$	involutivnost

4.2. Zakoni Booleove algebre

4.2.1. Zakon komutacije

Zakoni komutacije pokazuju da redoslijed dovođenja promjenjive ulazne veličine na ulaze logičkog sklopa nema utjecaja na rezultat logičke operacije.

$$A \cdot B = B \cdot A \quad \textbf{(27)}$$

$$A + B = B + A \quad \textbf{(28)}$$

4.2.2. Zakon asocijacije

Zakoni asocijacije pokazuju da način svrstavanja ulaznih veličina kod I i ILI operacija nema utjecaja na rezultat.

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C) \quad \textbf{(29)}$$

$$A + B + C = (A + B) + C = A + (B + C) \quad \textbf{(30)}$$

4.2.3. Zakon distribucije

Zakon distribucije pokazuje jednakost logičkog zbroja jedne varijable A sa svakim od članova B i C posebno. Ako u logičkom zbroju dvaju ili više članova postoji zajednički član, taj se član može izlučiti ispred zagrade.

$$A + B + C = (A + B) \cdot (A + C) \quad \textbf{(31)}$$

$$A \cdot (B + C) = A \cdot B + A \cdot C \quad \textbf{(32)}$$

4.3. De Morganovi teoremi

Među najvažnijim teoremima logičke algebre su i De Morganovi teoremi. Upotrebljavaju se vrlo često u postupku pojednostavljenja logičkih operacija. Teoremi upućuju na međusobnu dvojnost pojedinih logičkih operacija, tj. predstavljaju transformacijska pravila koja dozvoljavaju da se izrazi konjunkcije i disjunkcije mogu mijenjati jedan u drugi uz pomoć negacije te daju mogućnost realizacije logičkih sklopova pomoću NI i NILI sklopova.

Tablica 16. De Morganovi teoremi

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

Dodatni teoremi

$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

$$A \cdot B = \overline{\overline{A} + \overline{B}}$$

5. Složeni logički sklopovi

Složeni logički sklopovi sastoje se od više jednostavnih i ili izvedenih logičkih sklopova. Bez obzira na složenost logičkog sklopa uvijek je moguće napisati pripadajući logički algebarski izraz i sastaviti tablicu stanja. Dvije osnovne mogućnosti projektiranja složenih logičkih sklopova su:

- metoda logičkog umnoška logičkih zbrojeva
- umnožak maksterma.

Dobiveni algebarski izrazi za složene logičke operacije nisu uvijek minimalnog oblika pa se onda provodi i postupak minimizacije.

5.1. Minterm

Od kombinacija varijabla za koje funkcija ima logičku jedinicu dobije se zbroj minterma (*m*). S dvije ulazne varijable moguće je napraviti četiri različite operacije logičkih umnožaka. To su:

1) $\bar{A} \cdot \overline{B}$
2) $A \cdot \overline{B}$
3) $\bar{A} \cdot B$
4) $A \cdot B$.

Ako se za svaku od ovih operacija napravi tablica stanja za moguće vrijedosti varijabla na ulazu, vidi se da se na izlazu svaki put dobije samo jedna jedinica.

1. $Y = \bar{A} \cdot \overline{B}$

Tablica 17. Tablica stanja minterme 1)

A	B	$Y(m_0)$
0	0	1
1	0	0
0	1	0
1	0	0

2. $Y = A \cdot \overline{B}$

Tablica 18. Tablica stanja minterme 2)

A	B	$Y(m_1)$
0	0	0
1	0	1
0	1	0
0	0	0

3. $Y = \bar{A} \cdot B$

Tablica 19. Tablica stanja minterme 3)

A	B	Y(m_2)
0	0	0
1	0	0
0	1	1
1	1	0

4. $Y = A \cdot B$

Tablica 20. Tablica stanja minterme 4)

A	B	Y(m_3)
0	0	0
1	0	0
0	1	0
1	1	1

Ovakva logička operacija, koja na ulazu daje jedinicu samo za jednu ulaznu kombinaciju, a za sve ostale ulazne kombinacije na izlazu daje logičku nulu, naziva se minterm (*m*) jer je broj jedinica na izlazu minimalan.

Funkcija minterm može se ostvariti tako da se koristi operacija I, s time da se ulazi, koji su u stanju logičke 0 u kombinaciji što daju logičku 1, na izlazu prethodno invertiraju, tj. napravi se njihov komplement. S dvije ulazne varijable moguće je izvesti četiri različita minterma, tj. upravo onoliko koliki je broj različitih kombinacija s dvije varijable. To znači da s tri ulazne varijable postoji osam minterma, s četiri šesnaest itd.

Ako ste primijetili, u gornjim se tablicama često uz oznaku minterme (m) stavlja u indeks dekadski broj. Taj dekadski broj označava binarnu kombinaciju tog određenog broja.

Primjer 20.: Ako je minerm označen $m_{9,}$ to znači da 9 predstavlja binarni broj 1001, tj. za tu kombinaciju znamo da su prva i četvrta varijabla u logičkoj jedinici, a ostale varijable je potrebno invertirati. Isto tako, neki digitalni sklopovi u algebarskom zapisu su izraženi kao npr.:

$$Y = f(A,B,C,D) = \Sigma m\ (2,5,6,8,9). \quad \textbf{(33)}$$

Iz ovog zapisa možemo vidjeti da izraz Y ovisi o četiri varijable te da je funkcija izražena preko sume minterma, tj. m predstavlja minterme, a brojevi unutar zagrade njihove binarne kombinacije zapisane u obliku odgovarajućeg binarnog broja, tj. zapis unutar zagrade može biti:

$$f = m_2 + m_5 + m_6 + m_8 + m_{9.} \quad \textbf{(34)}$$

Tablica 21. Tablica stanja primjera 20

D	C	B	A	Y	*minerm*
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	m_2
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	m_5
0	1	1	0	1	m_6
0	1	1	1	0	
1	0	0	0	1	m_8
1	0	0	1	1	m_9
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	0	

Izveden izraz iz tablice stanja glasi:

$$Y = \Sigma m(2,5,6,8,9)$$

$$= m_2 + m_5 + m_6 + m_8 + m_9$$

$$= \overline{D} \cdot \overline{C} \cdot B \cdot \overline{A} + \overline{D} \cdot C \cdot \overline{B} \cdot A + \overline{D} \cdot C \cdot B \cdot \overline{A} + D \cdot \overline{C} \cdot \overline{B} \cdot \overline{A} + D \cdot \overline{C} \cdot \overline{B} \cdot A. \quad \textbf{(35)}$$

5.2. Maksterm

Maksterm (M) predstavlja logički zbroj kombinacija varijabli za koje funkcija ima logičku 0 na izlazu. S dvije ulazne varijable moguće je napraviti četiri različite operacije logičkog zbroja.

To su:

1) $A + B$
2) $\overline{A} + B$
3) $A + \overline{B}$
4) $\overline{A} + \overline{B}$.

Ako se za svaku od ovih operacija napravi tablica stanja za moguće vrijednosti varijabli na ulazu, onda se vidi da se u izlaznom stupcu svaki put dobije samo jedna nula.

1. $Y = A + B = M_0$

Tablica 22. Tablica stanja za maksterm 1)

B	A	$Y(M_0)$
0	0	0

0	1	1
1	0	1
1	1	1

2. $Y = \overline{A} + B = M_1$

Tablica 23. Tablica stanja za maksterm 2)

B	A	Y (M_1)
0	0	1
0	1	0
1	0	1
1	1	1

3. $Y = A + \overline{B} = M_2$

Tablica 24. Tablica stanja za maksterm 3)

B	A	Y(M_2)
0	0	1
0	1	1
1	0	0
1	1	1

4. $Y = \overline{A} + \overline{B} = M_3$

Tablica 25. Tablica stanja za maksterm 4)

B	A	Y(M_3)
0	0	1
0	1	1
1	0	1
1	1	0

Ovakva logička operacija koja na izlazu daje 0 samo za jednu ulaznu kombinaciju naziva se Makstrem (M), jer je broj jedinica na izlazu maksimalan. Makstrem se može ostvariti tako da se koristi operacija ILI, s time da one ulazne veličine, koje su u stanju 1 u kombinaciji, prethodno invertiraju, tj. da se napravi komplement. S dvije ulazne varijable moguće je izvesti četiri različita Makstrema, odnosno upravo onoliko koliki je broj različitih kombinacija s dvije ulazne varijable. Prema tome, s tri ulazne varijable postoji osam Makstrema, s četiri šesnaest Makstrema itd. Kao što ste primijetili, u gornjim tablicama često uz oznaku makstreme je i dekadski broj u indeksu. Taj dekadski broj označava binarnu kombinaciju određenog binarnog broja. Neki su digitalni sklopovi u algebarskom zapisu izraženi pomoću Makstrema npr.:

$Y = f(A, B, C, D) = \prod M(1,4,10,11,12).$ **(35)**

Iz ovog zapisa možemo vidjeti da izlaz Y ovisi o četiri varijable i da je funkcija izražena pomoću umnoška Makstrema:

$$f = M_1 + M_4 + M_{10} + M_{11} + M_{12}. \quad \textbf{(36)}$$

Primjer 21.:

$$Y = f(A, B, C, D) = \prod M(1, 4, 10, 11, 12) \quad \textbf{(37)}$$

Tablica 26. Tablica stanja primjera 21.

D	C	B	A	Y	Makstrem
0	0	0	0	1	
0	0	0	1	0	M_1
0	0	1	0	1	
0	0	1	1	1	
0	1	0	0	0	M_4
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	1	
1	0	0	1	1	
1	0	1	0	0	M_{10}
1	0	1	1	0	M_{11}
1	1	0	0	0	M_{12}
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

Iz tablice stanja možemo napisati algebarski izraz za primjer 21, koji glasi:

$$Y = (\bar{A} + B + C + D) \cdot (A + B + \bar{C} + D) \cdot (A + \overline{B} + C + \overline{D}) \cdot (\bar{A} + \overline{B} + C + \overline{D}) \cdot (A + B + \bar{C} + \overline{D}) \quad \textbf{(38)}$$

5.3. Univerzalnost logičkih sklopova NI i NILI

Svojstvo logičkih sklopova NI i NILI jest da se svaka osnovna, a prema tome i složena logička operacija, može ostvariti uporabom samo sklopova NI ili samo sklopova NILI. Svaka osnovna funkcija može se realizirati pomoću dovoljnog broja NI i NILI vrata, odnosno korištenjem De Morganovih pravila koji nam omogućuju pretvorbu proizvoljne funkcije u oblik izražen isključivo pomoću NI ili NILI vrata. Sklopovi NI i NILI višeulazni su sklopovi, odnosno nije specificirano koliki je minimalan ili maksimalan broj varijabla koje se na njih mogu istovremeno dovesti. Stoga ću u sljedećim primjerima koristiti NOR, što je za operatore NI, a za NILI sklop NAND, tj. operatori + i • bit će zamijenjen riječima NOR ili NAND. Iz toga je moguće realizirati ovakve funkcije prikazane u Tablica 27.

Tablica 27. Realizacija funkcija

$\overline{A}$ = NOR (A) = NAND (A)
$\overline{A + B} = NOR\ (A,B)$
$\overline{A \cdot B} = NAND\ (A,B)$
$\overline{A + B + C} = NOR\ (A,B,C)$
$\overline{A \cdot B \cdot C} = NAND\ (A,B,C)$

Nadovezivanje operatera moguće je izvesti još više različitih funkcija prikazanih u Tablica 28.

Tablica 28. Realizacija funkcija nadovezivanjem operatora

$A = \overline{\overline{A}} = NOR\ (NOR\ (A)) = NAND\ (NAND\ (A))$
$\overline{A + \overline{B + \overline{A + B}}} = NOR\ (NOR\ (A), NOR\ (B, NOR\ (A, B)))$

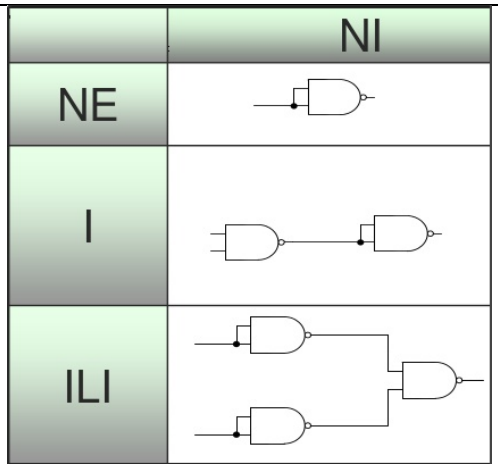
Važno je primijetiti sljedeća tri pravila koja će uvelike koristiti prilikom pretvaranja proizvoljne funkcije u oblik pogodan za realizaciju isključivo NI ili NILI sklopom:

1. Broj crta iznad varijabli označava negaciju koja odgovara broju NI ili NILI operatorima.
2. Svaki operator obuhvaća onaj „dio“ funkcije koji obuhvaća negacija.
3. Na mjestu gdje se u prikazu funkcija nalazi operator I ili ILI, u ovom se novom prikazu pomoću NOR i NAND nalazi zarez.

Ovakav je prikaz povoljan jer nam daje uvid koliko je NI ili NILI vrata potrebno za projektiranje određenog digitalnog sklopa.

5.3.1. Osnovne logičke funkcije pomoću NI logičkog sklopa

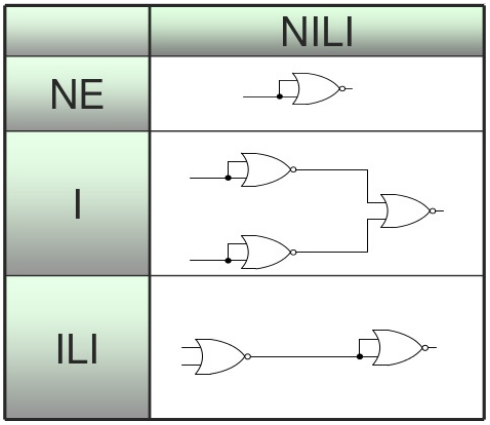
Kad se sklop NI upotrebljava kao sklop NE, koristi se samo jedan ulaz. Preostali ulazi moraju biti u stanju 1 ili kratko spojeni. Budući da sklop NI daje invertiranu operaciju I, ponovnim invertiranjem izlaz sklopa NI dobije se operacija I (tj. *NOR (NOR (A,B))*). Postupak realizacije sklopa ILI pomoću sklopa NI slijedi iz De Morganovih pravila. Ulazne veličine prvo se invertiraju (pomoću NI sklopa) i potom dovedu na ulaz NI sklopa.



Slika 29. Realizacija osnovnih logičkih sklopova pomoću NI vrata

5.3.2. Osnovne logičke funkcije pomoću NILI logičkog sklopa

Kada se sklop NILI upotrebljava kao sklop NE, koristi se jedan ulaz, a preostali ulazi moraju biti u stanju 0 ili kratko spojeni. Budući da sklop NILI daje invertiranu operaciju ILI, ponovnim invertiranjem pomoću NILI sklopa dobije se ILI operacija. Postupak za izvedbu sklopa I uporabom NILI sklopa slijedi iz De Morganovih pravila. Ulazne veličine prvo se invertiraju i onda dovedu na ulaz NILI sklopa.



Slika 30. Realizacija osnovnih logičkih operacija pomoću NILI vrata

6. Pretvorba proizvoljne funkcije u traženi oblik

Prvi korak kod pretvorbe proizvoljne funkcije jest minimizacija početne funkcije. To se može učiniti algebarski, K-tablicama ili Quine-McCluskey metodom. Pri minimalizaciji povoljno je minimaliziranu funkciju izraziti pomoću osnovnih logičkih elemenata (I, ILI, NE). Takav je koncept povoljan pri formuliranju problema i sagledavanju rješenja. Korištenjem Booleove algebre i transformacijskih pravila funkciju možemo izraziti pomoću samo NI sklopovima ili samo s NILI sklopovima. Ostvarenje digitalnog sklopa pomoću NI i NILI sklopa je povoljan jer je blizak električnoj izvedbi, tj. u izradi CMOS tehnologijom izrada NI i NILI vrata je jednostavna jer se koriste samo CMOS tranzistori, čija je konstrukcija jeftinija i jednostavnija. S obzirom na to da se traženi oblik može sastojati ili od isključivo NI operatora ili NILI operatora, razlikujemo dva osnovna slučaja kod pretvaranja:

1) Kod pretvaranja funkcije u oblik pogodan za realizaciju NI sklopovima nastoji se dvostrukom negacijom „razbiti“ sve I operacije.
2) Kod pretvaranja funkcije u oblik pogodan za realizaciju NILI sklopovima nastoji se dvostrukom negacijom „razbiti“ sve ILI operacije.

6.1. Minimizacija složenih logičkih funkcija

Logički sklopovi su elektronske komponente koji komplementiraju neke od složenih logičkih funkcija. Varijable funkcija su ulazi, a vrijednosti funkcija su izlazi logičkih sklopova. Logički sklopovi implementiraju potpune sisteme logičkih funkcija. Radi smanjenja troškova proizvodnje i složenosti digitalnog sistema teži se ispunjavanju određenih ciljeva, a neki od njih su:

- postići minimalno ostvarenje logičke funkcije ili tablice istinosti, tj. postići najjednostavniji sklop
- pojednostaviti izraz zbog tehnoloških i tehničkih razloga te zbog ekonomskih
- teži se što manjem broju različitih logičkih sklopova pa se često koristi samo NI ili NILI logički sklopovi.

U projektiranju digitalnih sklopova često su tehnički i ekonomski kriteriji kontradiktorni jedan drugome, a ponekada i navedeni ciljevi proturječe jedan drugome, tako da proces minimizacije podrazumijeva širi pojam nego samo pojednostavljenje logičkih izraza. To je

ujedno i proces optimizacije danog logičkog sklopa. Postoji više načina minimiziranja logičkih funkcija. Osnovni su:

1. Algebarska metoda ili algebarska transformacija
2. Karnaughove mape ili tablice, poznatije kao K – tablice
3. Quine–Mc Cluskeyeva metoda.

6.1.1. Algebarska metoda

Algebarski pristup minimizaciji složenih logičkih funkcija zasniva se na primjeni raznih pravila i teorema Booleove algebre. Primjenom raznih zakona uparivanja i zamjenom složenih podformula jednostavnijim, logički ekvivalentnim formulama pojednostavljujemo logički algebarski izraz i time se vrši minimizacija. Važno je napomenuti da se primjenom različitih teorema i Booleove algebre može doći do različitih minimiziranih izraza za istu funkciju, a da pritom imaju isti broj minimalnih logičkih elemenata.

Primjer 21.:

Treba minizirati izraz:

$$Y = \mathrm{A} \cdot \mathrm{C} \cdot (\bar{\mathrm{A}} + \mathrm{B}) + \mathrm{B} \cdot \bar{\mathrm{C}} \cdot (\mathrm{A} + \bar{\mathrm{B}}) \quad \textbf{(39)}$$

Tablica 29. Postupak minimizacije izraza Y u primjeru 21.

$A \cdot C \cdot \bar{\mathrm{A}} + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + B \cdot C \cdot \bar{B}$	Na dani izraz možemo primijeniti pravilo distributivnosti, tj. pomnožit ćemo članove izvan zagrade s članovima unutar zgrade i to za lijevi i desni dio izraza.
$A \cdot \bar{\mathrm{A}} \cdot C + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + B \cdot \bar{B} \cdot C$	Na krajnjem lijevom i krajnje desnom dijelu izraza članovima negirano A i negirano B zamijenit će mjesto radi jasnoće postupka (komutativnost).
$0 \cdot C + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + 0 \cdot C$	Sada je vidljivo da na ta dva dijela možemo primijeniti pravilo komplementarnosti (A pomnožen s negirano A i B s negirano B daje 0).
$0 + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + 0$	Za krajnji lijevi i krajnji desni dio izraza

	sada vrijedi anihilacija (množimo li nešto sa 0 dobiti ćemo 0).
$A \cdot C \cdot B + B \cdot \bar{C} \cdot A$	I te nule sada možemo „ispustiti“ iz izraza.
$A \cdot B \cdot (C + \bar{C})$	Sada možemo uočiti da su A i B zajednički članovi i lijevog i desnog gornjeg dijela izraza pa ih možemo izvući van (distributivnost).
A·B·1	Za C i negirano C unutar zagrade vrijedi komplementarnost.
A·B	Budući da množenje sa 1 ne mijenja na vrijednosti izraza, taj 1 se izostavlja iz izraza.

Uporaba NI i NILI sklopova na izraz:

f (A,B,C) = Y = A · B **(40)**

možemo primijetiti da zakon komplementarnosti vrijedi i za funkcije:

a) $f = \bar{\bar{f}} = \overline{\overline{A \cdot B}}$

NAND (NAND (A,B))

Iz ovog se izraza može vidjeti da je potrebno 2 NI sklopa za realizaciju.

b) $f = \bar{\bar{f}} = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}}$ (De Morganovo pravilo)

NOR (NOR (A), NOR (B))

Iz ovog se izraza može vidjeti da je potrebno 3 NILI sklopa za realizaciju izraza.

Primjer 22.:

Minimizirati sljedeću funkciju danu tablicom stanja.

Tablica 30. Tablica stanja primjera 22.

C	B	A	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f\ (A,B,C)\ =\ (\overline{A}\cdot B\cdot \overline{C})\ +\ (A\cdot B\cdot \overline{C})\ +\ (\overline{A}\cdot B\cdot C)$$

$$f\ =\ (\overline{A}\cdot B\cdot \overline{C})\ +\ (\overline{A}\cdot B\cdot C)\ +\ (A\cdot B\cdot \overline{C})\ +\ (\overline{A}\cdot B\cdot \overline{C})$$

$$f\ =\ (\overline{A}\cdot B)\cdot (\overline{C}+C)\ +\ (A+\overline{A})\cdot B\cdot \bar{C}$$

$$f\ =\ \overline{A}\cdot B\ +\ B\cdot \overline{C}$$

$$f\ =\ B\ +\ (\overline{A}\cdot \overline{C})$$

$$f\ =\ B\ +\ (\overline{A\ +\ C}) \qquad \textbf{(41)}$$

U ovom smo primjeru, nakon ispisivanja funkcije minterma, samo u drugom koraku dodali još jednu zagradu, tj. poduplali smo izraz (Ā·B·C) kako bismo bolje minimizirali funkciju i u zadnjem koraku primijenili De Morganovo pravilo.

Uporaba NI i NILI sklopova:

a) $\mathrm{f} = \bar{\bar{\mathrm{f}}} = \overline{\overline{\mathrm{B}\ +\ (\overline{\mathrm{A}\ +\ \mathrm{C}})}} = \overline{\overline{\mathrm{B}}\ \cdot\ (\overline{\mathrm{A}}\cdot\ \overline{\mathrm{C}})}$

NAND (NAND (B), NAND (NAND (A), NAND (C))

Iz ovog se izraza može vidjeti da je potrebno 5 NI sklopa za realizaciju ove funkcije.

b) $f\ =\ \overline{\overline{f}}\ =\ \overline{\overline{B\ +\ (\overline{A\ +\ C})}}$

NOR (NOR (B), NOR (A,C))

Iz ovog se izraza može vidjeti da je potrebno 3 NILI sklopa za realizaciju ove funkcije.

6.1.2. Karnaughove tablice (K-tablice)

Karnaughove tablice predstavljaju grafičku metodu minimalizacije logičkih funkcija. Koriste se za funkcije do 6 promjenjivih ulaznih veličina, a postaju nepregledne za više od 6 promjenjivih ulaznih varijabla. Ovu je metodu 1953. godine predstavio američki fizičar Maurice Karnaugh. Karnaughove tablice efikasna su metoda za pronalaženje pojednostavljenog algebarskog izraza. Slično tablici stanja, K-tablice sadrže polja u koja se upisuje stanje izlaza za određenu kombinaciju ulaznih varijabla. Ako je n broj promjenjivih ulaznih veličina, K-tablica sastoji se od 2^n polja. Stupci i redovi označavaju binarnu kombinaciju ulaznih varijabla i pravilo pri konstruiranju K-tablice je da razlike susjednih polja može biti samo u jednoj varijabli, tj. za jedno promjenjivo stanje. Zato se pri konstrukciji K-tablice koristi Grayev kod, tj. označavanje polja. Grayev kod piše se iznad

tablice, odnosno lijevo od tablice za određenu ili određene varijable. Na Slika 31. K-tablica primjera 23. prikazana je K-tablica. Valja primijetiti da su u polja K-tablice upisani dekadski brojevi. Ti brojevi predstavljaju binarni broj, tj. binarnu kombinaciju ulaznih varijabla za to polje. U polje K-tablice upisuje se „1“ ili „0“ ovisno o stanju izlazne funkcije za određenu kombinaciju ulaznih varijabla. K-tablice se mogu koristiti na dva načina. Jedan je način konstrukcija K-tablice pomoću tablice istinitosti određene funkcije, tj. jednostavnim upisivanjem jedinica u polja koja odgovaraju tablici istinosti, za koje je vrijednost funkcije 1 za dotičnu binarnu kombinaciju. Drugi način je konstruiranje K-tablice pomoću funkcije, tj. logička funkcija koja je zapisana u obliku logičke algebarske jednadžbe, a može se predstaviti pomoću K-tablice tako što se u svako polje mape upiše „1“ ako postoji konjunkcija u jednadžbi i to takva da je njena vrijednost 1 za vrijednost promjenjivih varijabli koje odgovaraju tom polju. Bolje rečeno, funkcija bi trebala imati oblik sume produkata. Minimalizacija pomoću K-tablice zasniva se na postupku tako da se grupiraju susjedna polja koja sadrže jedinice. Pri grupiranju polja važe sljedeća pravila:

- grupe se sastoje samo od jedinica
- broj jedinica u grupi mora biti potencija broja 2 ($1,2,4\ldots\ldots2^i$)
- jedinice moraju biti raspoređene u susjednim poljima kako bi ih mogli grupirati
- grupiraju se u pravokutnike (ne može ih se dijagonalno grupirati)
- svaka jedinica mora biti u nekoj grupi (jedna 1 samostalno se isto smatra grupom)
- grupe se mogu preklapati
- grupe, čija su polja u potpunosti sadržana u nekim drugim grupama, treba zanemariti
- smatra se da K-tablice imaju oblik valjka, odnosno mogu se grupirati i jedinice koje postaju susjedne kad se spoje - rubovi tablice (ili varijable). Gornja rubna polja mogu se spajati s donjim rubnim poljima, a to vrijedi i za desna i lijeva rubna polja.

Nakon što smo grupirali jedinice, gledamo varijablu (ili varijable) koja se mijenja u toj grupi, odnosno koja varijabla u toj grupi mijenja svoje stanje (iz „0“ u „ 1“ i obrnuto). Tu varijablu ispustimo iz te grupe, odnosno izraza. Osnovni princip koji garantira minimalnost je grupirati tako da se sa što manjim zaokruženjem obuhvati što više jedinica u K-tablici. Poslije

grupiranja jedinica, K-tablica se tumači kao disjunkcija konjunkcija koje odgovaraju grupama, a ne pojedinačnim jedinicama u tim grupama.

Primjer 23.:

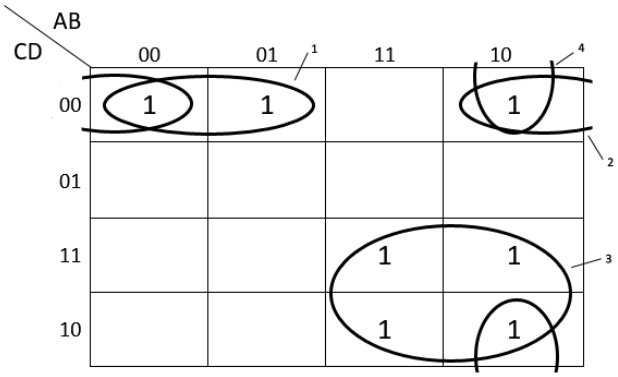
Minimiziraj funkciju pomoću K-tablice dane Tablica 31:

Tablica 31. Tablica stanja za primjer 23.

D	C	B	A	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Prvi korak u primjeni K-tablica predstavlja konstrukciju mintermi iz tablice istinosti (ovaj korak nije nužan). Sljedeći je korak upisivanje vrijednosti u K-tablicu u kojoj se susjedne kombinacije međusobno razlikuju za jedan, što je ključno za postupak minizacije. Treći korak u postupku minizacije jest grupiranje dviju, četiriju, osam jedinica zajedno ako je to moguće. Četvrti je korak eliminacija varijabla koje se mijenjaju. Zadnji korak jest zbrajanje varijabla koje su ostale.

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot C \cdot \bar{D} + A \cdot B \cdot C \cdot D + A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} \quad \textbf{(42)}$$



Slika 31. K-tablica primjera 23.

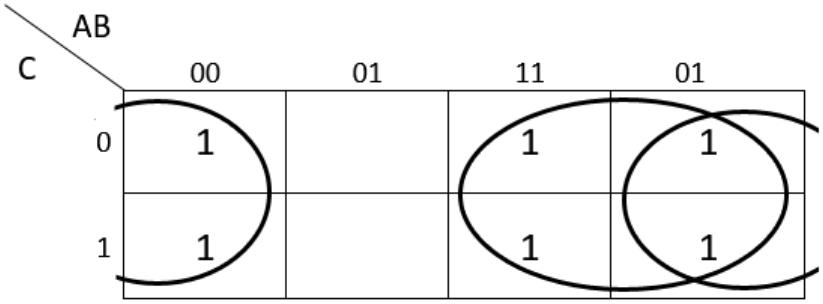
$$Y = \overline{A} \cdot \bar{C} \cdot \bar{D} + \bar{B} \cdot \bar{C} \cdot \bar{D} + A \cdot C + A \cdot \bar{B} \cdot \bar{D} \quad \textbf{(43)}$$

Primjer 24.:

Minimiziraj funkciju pomoću K-tablice dane algebarskom funkcijom.

$$f\,(A,B,C) = A \cdot \bar{C} + A \cdot B \cdot C + A \cdot \bar{B} \cdot C + \overline{A+B} \quad \textbf{(44)}$$

Ako je funkcija zadana kao suma mintermi ili makstermi, funkciju možemo direktno u K-tablicu upisati. Ako to nije ispunjeno, treba se težiti tome da funkciju treba zapisati u obliku sume produkata, koji nisu nužno mintermi. U ovom se primjeru analizira član po član. Vidimo da član A·C nije minterm, stoga možemo zaključiti da je funkcija jednaka „1“ ako je A u jedinici a C u nuli, neovisno o varijabli B. Pogledom u tablicu pronalazimo polja za koja su umnožak A i C komplementa jednaka 1 i u ta polja upisujemo „1“. Isto tako, ako primijetimo, zadnji član pomoću De Morganovih pravila možemo zapisati kao produkt A·B komplementa i na isti način naći polja za koja vrijedi da daju jedinicu i za ta polja upisati „1“. Ostali članovi su mintermi.



Slika 32. K-tablica primjera 24.

Rješenje glasi:

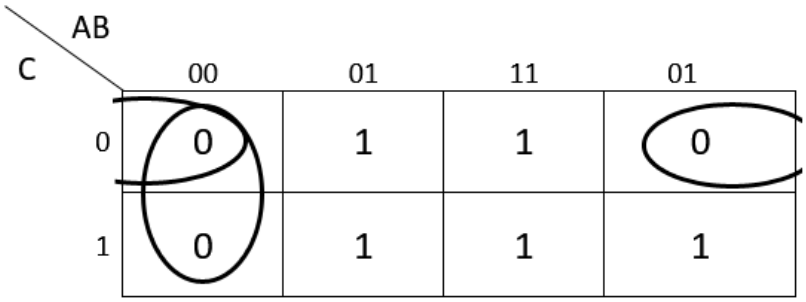
$$f = A + \bar{B}. \quad \textbf{(45)}$$

Primjer 25.: Isto tako K-tablice mogu služiti za minimizaciju pomoću maksterme, odnosno pomoću „0“. Minimiziraj funkciju dane tablicom stanja.

Tablica 32. Tablica stanja primjera 25.

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$Y = \prod M\ (0,2,4)$ **(46)**



Slika 33. K-tablica primjera 25.

Rješenje dano K-tablicom glasi:

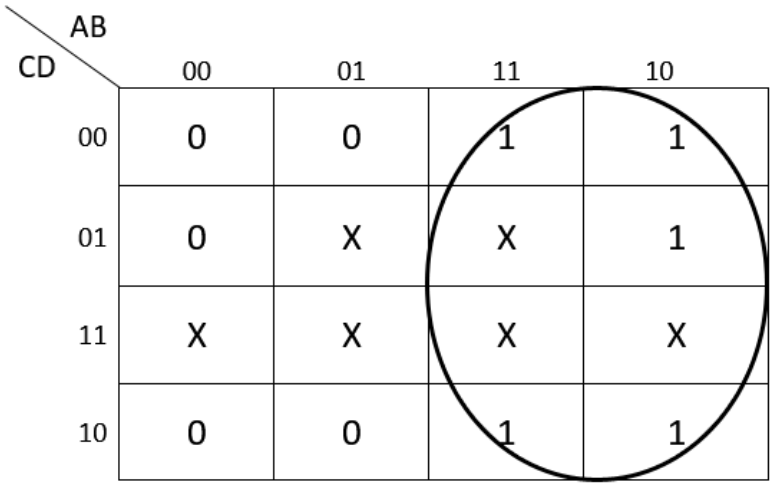
$$Y = \left(\overline{A} + \overline{B}\right) \cdot \left(\overline{C} + \overline{A}\right). \quad \textbf{(47)}$$

6.1.2.1. Nepotpune specifične funkcije

Nepotpune specifične funkcije su logičke funkcije koje nisu u potpunosti definirane za sve logičke kombinacije na ulazu. Postoje dva moguća tumačenja ove definicije. Jedan je da će na ulazu, tj. da će ulazne varijable tvoriti binarnu kombinaciju za koju izlazna funkcija ne mari, popularno zvanoj „*don't care*" i vrijednost funkcije za tu kombinaciju se označuje sa X. Drugi način tumačenja je da se ta određena kombinacija ulaznih varijabli nikada neće dogoditi, popularno nazvana „*can't happen*" i ona se isto označuje sa X. Pri minizaciji pomoću K-tablica, ti X-evi se upisuju u tablicu i jako su korisni jer ih možemo tumačiti kako nam odgovara, tj. interpretira se kako najbolje odgovara minimizaciji (da li kao logička „1" ili logička „0") i time postići što veće zaokruživanje.

Primjer 26.:

Uzmimo za primjer BCD kod. Binarne kombinacije koje kod koristi odgovaraju binarnim kombinacijama za brojeve od 0 do 9, a ostale se kombinacije ne koriste i možemo ih pridodati vrijednosti X. Ako bismo željeli napraviti funkciju koja će nam govoriti kad je izlazna varijabla neparni broj, možemo pomoću K-tablica napraviti minimizaciju te funkcije.



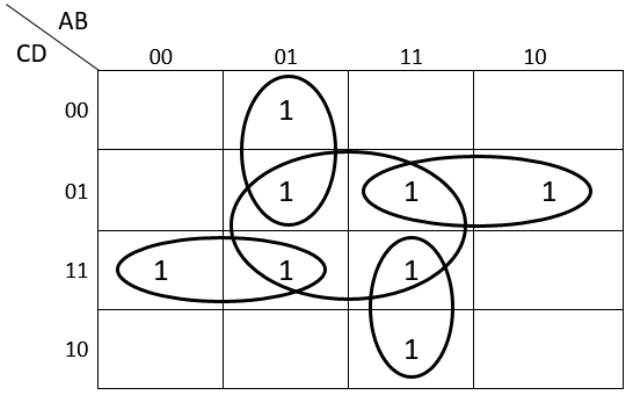
Slika 34. K-tablica s „don't care“ vrijednostima

Ono što primjećujemo jest da možemo napraviti dvije grupe, jednu od četiriju jedinica, drugu od dviju, no ako protumačimo vrijednost X, koji su u susjednim poljima, kao jedinicu, primjećujemo da možemo zaokružiti sve jedinice u jednu grupu i time dobiti rješenje:

f = A. **(48)**

6.1.2.2. Redundantna zaokruživanja

Uzmimo za primjer sljedeću K-tablicu danu Slika 35.



Slika 35. K-tablica s redundatnim zaokruživanjem

Minimiziramo pomoću K-tablice i dobijemo sljedeći izraz:

$$Y = BD + ABC + \overline{A}CD + \overline{A}B\overline{C} + A\overline{C}D. \quad \textbf{(49)}$$

Sada analizirajmo malo bolje sva zaokruženja koja smo napravili i prisjetimo se osnovne ideje minimizacije. Osnovna ideja minimizacije jest pokriti sve jedinice, tako da se u svakom

zaokruženju nalazi barem jedan minterm koji ne pokriva nijedno drugo zaokruženje. No, kao što je prikazano, imamo 5 zaokruženja. Najveće zahvaća 4 jedinice, a svi ostali po 2 jedinice. Ako promotrimo, vidimo da unutar velikog zaokruženja svaka jedinica pripada, tj. nalazi se u manjem zaokruženju. Time veliko zaokruženje postaje nepotrebno jer sve jedinice unutar toga zaokruženja imaju svoje grupe. To je nepotrebno zaokruženje te je redudantno. Razlog zašto je najveće zaokruženje redundantno jest taj što svaka od manjih zaokruženja pokriva jedinice koje nijedno drugo zaokruženje ne pokriva, a veliko zaokruženje pokriva jedinice koje ostala manja zaokruženja pokrivaju. Time je rješenje:

$$Y = ABC + \overline{A}CD + \overline{A}B\overline{C} + A\overline{C}D. \quad \textbf{(50)}$$

Ovakve su pogreške česte pa je zbog toga potrebno nakon grupiranja jedinica provjeriti dva uvjeta:

1) *uvjet mintermnosti*:

   svako zaokruženje koje ulazi u minimalni oblik funkcije mora imati barem jedan minterm koji je pokriven isključivo njime

2) *uvjet redundantosti*:

   zaokruženje čiji je svaki minterm pokriven još nekim zaokruženjem jest redundantno.

6.1.3. Quine-McCluskeyeva metoda

Quine-McCluskeyeva metoda dobila je ime po svojim izumiteljima. Metodu je Willard Van Orman Quine predstavio 1952. godine, a kasnije 1956. godine ju je poboljšao Edward J. McCluskey koji je razvio algoritam za minimalizaciju složenih logičkih sklopova. Ova je metoda za razliku od K-tablice (grafička metoda) tablična metoda koja je prikladna za minimizaciju većeg broja varijabla. Budući da je ova metoda numerička metoda, prikladna je za programsku implementaciju. Isto tako može se svesti na manipuliranje indeksom standardnih članova. U Quine-McCluskey metodi se koristi pravilo iz Booleove algebre (34**)** koje za ovu metodu glasi:

$$A \cdot B + \overline{A} \cdot B = (A + \overline{A}) \cdot B \quad \textbf{(51)}$$

$$\left(A + \overline{A}\right) \cdot B = 1 \cdot B = B. \quad \textbf{(52)}$$

Prije objašnjenja metode bitno je napomenuti dva pojma, a to su:

- *implikant* je svaki produkt (ulaznih varijabla) koji daje vrijednost funkcije „1“, tj. logičku jedinicu. U K-tablici to su sva moguća zaokruženja.
- *primarni implikant* su oni implikati koji nisu u cijelosti sadržani ni u jednom drugom implikantu.

Metoda ima dvije faze, gdje je prva faza nalaženje primarnih implikanata (najvećih zaokruženja u K-tablici), a druga je faza određivanje optimalnog (minimalnog) skupa primarnih implikanata.

Opis metode:

1. korak:

   - ispisivanje binarnih kombinacija minetrama, tj. prikazati njihov binarni zapis. Isto tako ako ima „don’t care“ njih isto ispisati jedan ispod drugoga.

2. korak:

   - formiranje tablica minterma i „don’t care“ (ako ih ima) tako da ih grupiramo po broju jedinica. To znači da prvu grupu sačinjavaju kombinacije s nula „1“, drugu grupu sačinjavaju kombinacije s jednom „1“, treću grupu kombinacije s dvije „1“, itd. Prvo idu mintermi, jedan ispod drugog, onda „*don’t care*“ kombinacije. Uz svaku kombinaciju stavljamo zastavicu „D“ (ako nemamo „*don’t care*“, zastavica nam ne treba).

3. korak:

   - u njemu krećemo kombinirati indekse (binarne kombinacije minterma i „*don’t care*“) iz jedne grupe sa susjednom grupom i to tako da uzmemo jednu kombinaciju iz niže grupe i pokušamo je kombinirati s indeksom u gornjoj grupi. Indekse možemo kombinirati samo ako se ne razlikuju u jednoj varijabli i na mjestu gdje se u binarnoj kombinaciji razlikuju ta dva indeksa pa stavljamo oznaku X (često se koristi _ ili d) i tako stavljamo taj novi indeks u novi stupac tablice i pokraj u zagradu upisujemo koje kombinacije smo kombinirali. U tablici isto tako označujemo (*) koje smo indekse kombinirali. Tako redom kombiniramo indekse dok nismo napravili novu tablicu i zahvatili sve indekse iz prve tablice. Napomena: Možemo kombinirati samo indekse koji su iz susjednih grupa. Ako kombiniramo dva „*don’t care*“, zastavicu „D“ stavljamo u jedinicu.

4. korak:

   - nakon što smo formirali drugi stupac, moramo ga podijeliti u grupe i to na način da svi indeksi, koji su dobiveni grupacijom prve i druge grupe, sada su prva grupa, druga grupa su indeksi dobiveni grupacijom druge i treće grupe itd. Sada opet primjenjujemo postupak iz prethodnog koraka, s time da možemo upariti one indekse gdje je „X“ oznaka na istom mjestu, tj. poziciji i moraju se samo u jednoj varijabli razlikovati. Opet stvaramo novi stupac u koji pišemo taj dobiveni indeks s time da će sada imati dvije oznake „X“. Indekse sa zastavicom „D : 1“ ne kombiniramo.

5. korak:

   - ponovimo 4. korak za novi stupac ako je moguće. Nakon toga označimo sve indekse (produkte) koji su ostali nekombinirani. To su implikanti i njih zovemo primarni implikanti.

6. korak:

   - konstruiramo novu tablicu gdje su stupci mintermne funkcije. Iznad stupaca pišemo odgovarajući dekadski zapis binarnog broja određene binarne kombinacije. U retke pišemo vrijednosti produkta koji su označeni u prethodnom koraku i lijevo od njih pišemo njihov algebarski oblik. U tablicu stavljamo „X“ na mjesto gdje se produkt križa sa svojom mintermom („don’t care“ vrijednosti ne pišemo u tablicu), tj. gledamo brojeve u zagradama indeksa (produkta) i križamo ih s odgovarajućim stupcem.

7. korak:

   - sada kada smo konstruirali tablicu, gledamo koji redak pokriva najviše stupaca i koji redak pokriva stupce koje samo on pokriva, tj. on jedini pokriva određeni minterm i desno od tablice, dajemo kvačicu za one retke koji to ispunjavaju (to su primarni implikanti).

8. korak:

   - ako nismo pokrili sve minterme, prelazimo u sljedeći korak korištenjem Pyne-McCluskey pristupa. Cilj nam je pokriti sve minterme, ali pritom moramo paziti na minimalni oblik funkcije.

   Pyne-McCluskey pristup:

   - Svakom produktu (retku) u tablici damo desno od tablice neku oznaku P_n (n broj $n \in [0,\infty]$).

- Isključujemo iz postupka onaj redak za koji vrijedi korak 7. tj. te minterme koji pripadaju pod taj redak ne uzimamo u ovom pristupu.
- Sada kombiniramo koristeći Booleovu algebru i to na način da disjunkcijom „+“ uparimo sve P_n kombinacije koje pokrivaju određeni minterm i te P_n -ove spojimo konjunkcijom „·“ te izmnožimo zagrade i vodimo računa da radimo s Booleovom algebrom.
- Sada gledamo kada je funkcija u jedinici i gledamo članove s manjim P_n brojem članova. Treba imati na umu da postoji više rješenja i time smo dobili popis „kandidata“ za minimalnu funkciju.

9. korak:

   - sada kombiniramo „kandidate“ iz prethodnog koraka i retke iz 7. koraka. Ispisujemo algebarski izraz tih kombinacija i dobivamo onu kombinaciju koja nam najviše odgovara.

Primjer 26.:

Minimizirati funkciju Quine-McCluskey metodom. Funkcija je zadana u sljedećem obliku:

$$f(A,B,C,D) = \Sigma m(0,4,5,7,8,9,13) + \Sigma d(2,3,12,15) \quad \textbf{(53)}$$

ispisujemo binarne kombinacije brojeva i lijevo od njih pišemo odgovarajući dekadski broj pa ih grupiramo u grupe (G1, G2, G3, G4, G5).

Tablica 33. Quine-McCluskeyeva tablica primjera 26.

	A B C D				
0	0 0 0 0	**G1**	0000 * D : 0 (0)	00X0 D : 0 (0,2)	XX00 (0,4,8,12)
4	0 1 0 0	**G2**	0010 * D : 1 (2)	0X00 * D : 0 (0,4)	XX00 (0,8,4,12)
5	0 1 0 1	**G2**	0100 * D : 0 (4)	X000 * D : 0 (0,8)	X10X (4,5,12,13)
7	0 1 1 1	**G2**	1000 * D : 1 (8)	001X D : 1 (2,3)	1X0X (8,9,12,13)
8	1 0 0 0	**G3**	0011 * D : 1 (3)	010X * D : 0 (4,5)	X1X1 (5,7,15,13)
9	1 0 0 1	**G3**	0101 * D : 0 (5)	X100 D : 0 (4,12)	X1X1 (7,13,5,13)
13	1 1 0 1	**G3**	1001 * D : 0 (9)	100X * D : 0 (8,9)	
2	0 0 1 0	**G3**	1100 * D : 1 (12)	1X00 D : 0 (8,12)	
3	0 0 1 1	**G4**	0111 * D : 0 (7)	0X11 D : 0 (3,7)	
12	1 1 0 0	**G4**	1101 * D : 0 (13)	01X1 * D : 0 (5,7)	
15	1 1 1 1	**G5**	1111 * D : 1 (15)	X101 * D : 0 (5.13)	
				1X01 D : 0 (9,13)	
				110X * D : 0 (7,13)	
				11X1 * D : 0 (15,13)	

Nadalje sada kombiniramo indekse. Uzet ćemo primjer iz grupe G2, indeks (8) i iz grupe G3, indeks (12). Prvo gledamo razlikuju li se kombinacije u samo jednoj varijabli. Odgovor je DA, razlikuju se samo u jednoj varijabli na drugom mjestu. Sada u novom stupcu pišemo tu binarnu kombinaciju, samo što na tom drugom mjestu stavljamo X i u zagradi pišemo (8,12).

To znači da se na tom mjestu razlikuju. Zastavicu pišemo D : 0, sve dok u kombinaciji se nalazi minterm, zastavica će biti "0". Sada taj postupak ponovimo za drugi stupac i uvidimo da prvi indeks u stupcu ne možemo kombinirati, pa pokraj njega stavljamo (*) zvjezdicu. Moramo paziti da kombiniramo indekse koji na istom mjestu imaju „X". Ako nemaju, možemo ih kombinirati s indeksom (12,13), iako na istoj poziciji imaju „X", jer se razlikuju u tri varijable. Sada gledamo treći stupac i može li se isto kombinirati. Nakon toga, prelazimo u Pyne-McCluskey pristup. Primjećujete da imamo duplikate, a njih tretiramo kao da piše samo jedan od njih.

Tablica 34. Pyne-McCluskeyeva tablica

		0	4	5	7	8	9	13	
$\bar{A} \cdot \bar{B} \cdot \bar{D}$	(0,2)	X							P_0
$B \cdot \overline{C} \cdot \bar{D}$	(4,12)		X						P_1
$A \cdot \overline{C} \cdot \bar{D}$	(8,12)					X			P_2
$\bar{A} \cdot C \cdot D$	(3,7)				X				P_3
$A \cdot \overline{C} \cdot D$	(9,13)						X	X	P_4
$\overline{C} \cdot \overline{D}$	(0,4,8,12)	X	X			X			P_5
$B \cdot \overline{C}$	(4,5,12,13)		X	X				X	P_6
$A \cdot \overline{C}$	(8,9,12,13)					X	X	X	P_7
$B \cdot D$	(5,7,15,13)			X	X			X	P_8

Kao što vidimo, one indekse koje nismo kombinirali od njih sada radimo tablicu. Ako primijetimo, u drugom stupcu u ovu tablicu nismo uvrstili indeks (2,3) jer imaju zastavicu u jedinici. Isto tako svaki minterm pokriva barem dva produkta, tako da nemamo redak koji zadovoljava kriterije da jedan redak pokriva samo jedan minterm. Pišemo funkciju P:

$$P = (P_0+P_5) \cdot (P_1+P_5+P_6) \cdot (P_6+P_8) \cdot (P_3+P_8) \cdot (P_2+P_5+P_7) \cdot (P_4+P_7) \cdot (P_4+P_6+P_7+P_8)$$

$$P = (P_0P_1+P_0P_5+P_0P_6+P_1P_5+P_5+P_5P_6) \cdot (P_3P_6+P_6P_8+P_3P_8+P_8) \cdot (P_4+P_6+P_7+P_8) \cdot (P_2P_4+P_2P_7+P_4P_5+P_5P_7+P_4P_7+P_7).$$

Radi jednostavnijeg zapisa pisat ćemo samo indekse

$$P = (P_{0,1,3,6} + P_{0,1,6,8} + P_{0,1,3,8} + P_{0,1,8} + P_{0,3,5,6} + P_{0,5,6,8} + P_{0,3,5,8} + P_{0,5,8} + P_{0,3,6} + P_{0,6,8} + P_{1,3,5,6} + P_{1,5,6,8} + P_{1,3,5,8} + P_{1,5,8} + P_{3,5,6} + P_{5,6,8} + P_{3,5,8} + P_{5,8} + P_{3,5,6} + P_{5,6,8} + P_{3,5,6,8} + P_{5,6,8}) \cdot (P_{2,4} + P_{2,4,6} + P_{2,4,7} + P_{2,4,8} + P_{2,4,7} + P_{2,6,7} + P_{2,7} + P_{2,7,8} + P_{4,5} + P_{4,5,6} + P_{4,5,7} + P_{4,5,8} + P_{4,5,7} + P_{5,6,7} + P_{5,7} + P_{5,7,8} + P_{4,7} + P_{4,6,7} + P_{4,7} + P_{4,7,8} + P_{4,7} + P_{6,7} + P_7 + P_{7,8})$$

$$P = (P_{0,1,8} + P_{0,3,6} + P_{0,5,8} + P_{0,6,8} + P_{0,1,3,6} + P_{0,1,3,6,8} + P_{0,3,5,6} + P_{0,5,6,8} + P_{0,3,5,8} + P_{0,3,6,8} + P_{1,5,8} + P_{1,3,5,6} + P_{1,5,6,8} + P_{1,3,5,8} + P_{3,5,6} + P_{3,5,8} + P_{3,5,6,8} + P_{5,8} + P_{5,6,8}) \cdot (P_{2,4} + P_{2,7} +$$

$P_{2,4,6} + P_{2,4,7} + P_{2,4,8} + P_{2,6,7} + P_{2,7,8} + P_{4,5} + P_{4,7} + P_{4,5,6} + P_{4,5,7} + P_{4,5,8} + P_{4,6,7} + P_{4,7,8}$

$+ P_{5,7} + P_{5,6,7} + P_{5,6,7} + P_{5,6,8} + P_{6,7} + P_7 + P_{7,8})$

$P = P_{0,1,2,4,8} + P_{0,1,2,7,8} + P_{0,1,2,4,6,8} + P_{0,1,2,4,7,8} + P_{0,1,2,7,8} + P_{0,1,2,4,6,7} + P_{0,1,2,7,8} + P_{0,1,4,5,8} +$

$P_{0,1,4,7,8} + P_{0,1,4,5,6,8} + P_{0,1,4,5,7,8} + P_{0,1,4,5,8} + P_{0,1,4,6,7,8} + P_{0,1,4,7,8} + P_{0,1,5,7,8} + P_{0,1,5,6,7,8} +$

$P_{0,1,5,7,8} + P_{0,1,6,7,8} + P_{0,1,7,8} + P_{0,2,3,4,6} + P_{0,2,3,6,7} + P_{0,2,3,4,6} + P_{0,2,3,4,6,7} + P_{0,2,3,4,6,8} +$

$P_{0,2,3,6,7} + P_{0,2,3,6,7,8} + P_{0,3,4,5,6} + P_{5,7,8} + P_{5,6,7,8} + P_{4,5,6,8} + P_{5,6,7,8} + \ldots\ldots$

$P= P_{4,5,8} + P_{5,7,8} + P_{0,1,7,8} + P_{0,3,6,7} + \ldots\ldots\ldots + P_{4,5,6,8} + P_{5,6,7,8} + \ldots + P_{0,2,3,6,7} + \ldots\ldots$

(u minimalni oblik uzimamo one umnoške sa što manjim brojem P-članova)

$$P_{min} = P_4 \cdot P_5 \cdot P_8 + P_5 \cdot P_7 \cdot P_8. \quad \textbf{(54)}$$

Vidimo da je funkcija poprimila vrijednost 1 ako je bilo koji od izraza za sumu produkata P_n u jedinici. Sada tražimo produkte koji imaju što manje P-članova. Time dobivamo da taj uvjet zadovoljavaju gornji članovi, tj. produkti.

Minimalni oblik funkcije jest :

a) $P_4,P_5,P_8 \rightarrow f = A \cdot \overline{C} \cdot D + \overline{C} \cdot \overline{D} + B \cdot D$ **(55)**

b) $P_6,P_7,P_8 \rightarrow f = B \cdot \overline{C} + A \cdot \overline{C} + B \cdot D.$ **(56)**

Izbor minimalnog oblika zadane funkcije ovisi o puno faktora. Npr., ako bismo tražili minimalni oblik s minimalnim brojem Booleovih operacija, onda bi izraz b) bio bolji odabir nego izraz a).

Primjer 27.:

Uzmimo da u gornjem primjeru nemamo u tablici redak P_3, time redak P_8 odgovara 7. koraku objašnjenja Quine-McCluskeyjeve metode. To znači da bismo pokraj P_8 stavili kvačicu, tj. oznaku da on zadovoljava taj uvjet i u zadnjem bismo retku stavili kvačice na one minterme koje P_8 pokriva pa bismo tako pokrivali mintermu 5,7,13. U izrazu P ne bismo uzimali te minterme nego one ispod kojih nisu kvačice, tj. minterme 0,4,8,9. Time bismo dobili sljedeći izraz:

$$P = (P_0+P_5) \cdot (P_1+P_5+P_6) \cdot (P_2+P_5+P_7) \cdot (P_4+P_7). \quad \textbf{(57)}$$

Na kraju, u minimalni izraz kombinira se P_8 s minimalnim produktima sumama izraza P.

Tablica 35. Pyne-McCluskeyeva tablica za primjer 27

0	4	5	7	8	9	13	
X							P_0
	X						P_1
				X			P_2
					X	X	P_4
X	X			X			P_5
	X	X				X	P_6
				X	X	X	P_7
		X	X			X	P_8 ✓

$P = P_5 \cdot P_7 + P_6 \cdot P_7 + \ldots\ldots$

a) $P_8, P_5, P_7 \rightarrow$ $$f = B \cdot D + \overline{C} \cdot \overline{D} + A \cdot \overline{C} \quad \textbf{(58)}$$

b) $P_8, P_6, P_7 \rightarrow$ $$f = B \cdot D + B \cdot \overline{C} + A \cdot \overline{C} \quad \textbf{(59)}$$

7. Potpuno zbrajalo

U današnje je vrijeme zbrajalo sveprisutni logički sklop koji je integriran u mnoge uređaje i aparate. Zbrajalo je složeni logički sklop koji obavlja funkciju zbrajanja binarnih brojeva. Potpuno zbrajalo sastavni je dio aritmetičko-logičke jedinice (ALU). Često se naziva i binarno zbrajalo. Osim što sudjeluje u operaciji zbrajanja binarnih brojeva, sudjeluje u još mnogo drugih logičko-aritmetičkih operacija, kao što je računanje adrese, sudjeluje u operaciji množenja, operator dodavanja inkrementa itd. Potpuno zbrajalo može biti konstruirano da zbraja binarne brojeve u raznim kodovima (kao što je EXCESS-3, Grayev kod itd.). Sastavni dio zbrajala je polu-zbrajalo, odnosno potpuno zbrajalo sastoji se od dvaju polu-zbrajala povezanih u cjelinu.

7.1. Polu-zbrajalo

Polu-zbrajalo je složeni logički sklop, s dva ulaza i s dva izlaza. Obavlja logičku operaciju zbrajanja dvaju jednobitnih binarnih brojeva. Ako se prisjetimo tablice zbrajanja iz prehodnog poglavlja 3.2.1, uvidjet ćemo da tablica zbrajanja odgovara tablici stanja za EX–ILI logički sklop, dok tablica stanja za *carry bit* odgovara tablici stanja za I–logički sklop. Iz toga vidimo da je polu-zbrajalo složeni logički sklop koji se sastoji od EX–ILI i I-logičkih vrata, prikazanih na Slika 36Error! Reference source not found.. Ulazi polu-zbrajala su dva binarna broja, u ovom slučaju A i B, dok su izlazi zbroj ta dva binarna broja (S) i *carry bit* (C), tj. izlaz koji ukazuje je li došlo do preljeva jedinice. Za polu-zbrajalo kažemo da je nepotpuno zbrajalo, jer može zbrojiti dva binarna broja koja su jednobitna ili dva bita, koja se nalaze na najmanjem težinskom mjestu kod višebitnih brojeva. Ukoliko dođe do preljeva jedinice, utoliko polu-zbrajalo neće taj preljev uzimati u obzir jer ga „ne vidi“ i time dolazi do pogreške u zbrajanju višebitnih binarnih brojeva. To znači da proces zbrajanja dvaju binarnih brojeva nije potpun, nego djelomičan pa se zato ovo zbrajalo naziva polu-zbrajalo (eng. *Half–adder*). Shema polu-zbrajala i simbol uz tablicu stanja je prikazan:

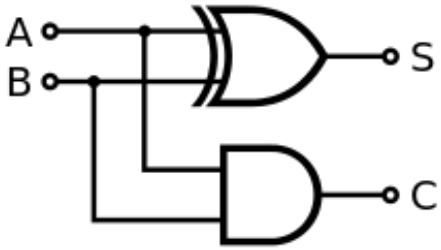
Tablica 36. Tablica stanja polu-zbrajala

A	B	C	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

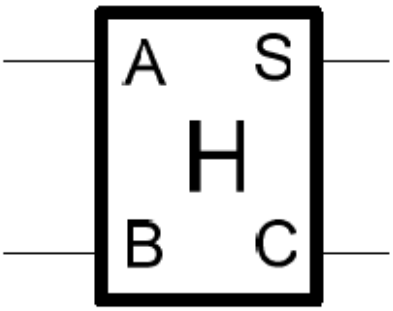
Algebarski izraz:

$$S = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B} \quad \textbf{(60)}$$

$$C = A \cdot B. \quad \textbf{(61)}$$



Slika 36. Shema polu-zbrajala



Slika 37. Simbol polu-zbrajala

7.2. Zbrajalo

Zbrajalo je složeni logički sklop koji zbraja binarne brojeve i uzima u obzir *carry bit* te se za njega koristi termin potpuno zbrajalo. Danas je potpuno zbrajalo integrirano skoro u sve digitalne sklopove pa je rijetkost da ga se susreće kao poseban sklop. Premda se u današnje vrijeme na tržištu nudi kao zaseban čip, varijacije su u tipu zbrajala i veličini binarnih brojeva koje može zbrajati. Općenito, zbrajalo (jednobitni broj) ima tri ulaza i dva izlaza, gdje su dva ulaza, ulazi binarnih brojeva (A,B), a treći ulaz je za *carry bit* (C_{in}), odnosno za signal je li došlo do preljeva. Izlazi su rezultati binarnog zbrajanja i signala koji govori je li došlo do preljeva jedinice (C_{out}). Najčešća izvedba zbrajala je serijskim spajanjem dvaju poluzbrajala, tj. kaskadiranjem polu-zbrajala, a *carry* signali se dovode na ILI vrata koja daju na izlazu carry signal zbrajala. Naravno, moguće su i druge izvedbe zbrajala. Budući da je zbrajalo složeni logički sklop, prvo treba zadovoljiti tablicu stanja (Tablica 37. Tablica stanja potpunog zbrajala), a unutarnjom se izvedbom može razlikovati ovisno

o zahtjevima. Zbrajalo zbraja dva binarna broja i *carry* signal, a na izlazu daje njihov zbroj. Na Slika 40. prikazana je logička shema i simbol zbrajala (engl. *Full-adder*).

Tablica 37. Tablica stanja potpunog zbrajala

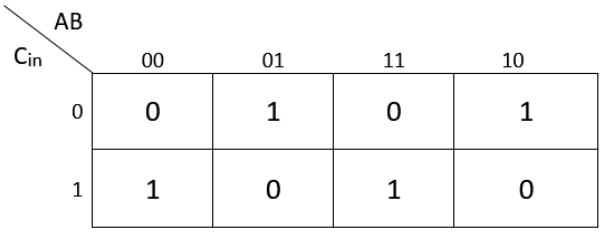
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Algebarski izraz za potuno zbrajalo:

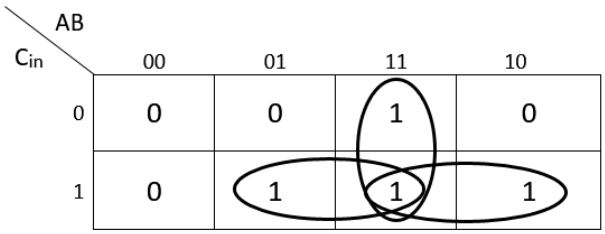
$$\begin{aligned} S &= Cin \oplus (A \oplus B) = \overline{Cin}\left(A\overline{B} + \overline{A}B\right) + Cin\left(\overline{A\overline{B} + \overline{A}B}\right) \\ &= \overline{Cin}\left(A\overline{B} + \overline{A}B\right) + Cin\left(AB + \overline{A\overline{B}}\right) \\ &= AB\overline{Cin} + \overline{A}B\overline{Cin} + ABCin + \overline{A\overline{B}}Cin \end{aligned} \quad \textbf{(62)}$$

$$C_{out} = Cin\left(A\overline{B} + \overline{A}B\right) + AB = A\overline{B}Cin + \overline{A}BCin + AB. \quad \textbf{(63)}$$

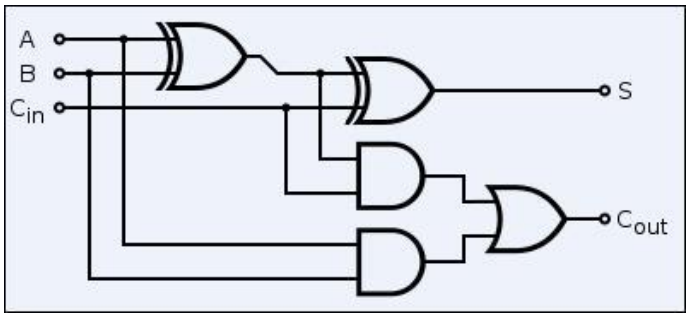
Ako pogledamo K-tablice za izlaz S i C_{out}, uvidjet ćemo da ovi izrazi isto vrijede i za K-tablice.



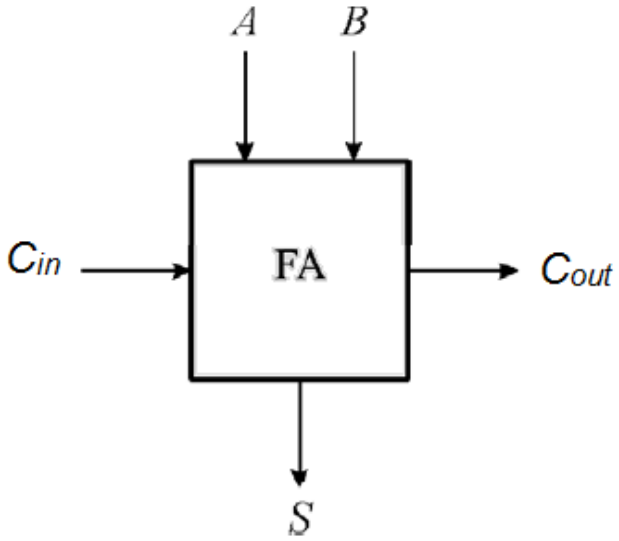
Slika 38. K-tablica za izlaz S



Slika 39. K-tablica za izlaz C_{out}



Slika 40. Shematski prikaz potpunog zbrajala



Slika 41. Simbol potpunog zbrajala

7.2.1. Izvedba potpunog zbrajala pomoću NI vrata

Budući da je zbrajalo složeni logički sklop, možemo ga realizirati korištenjem samo NI vratima. Prvo analiziramo izlaze zbrajala, imamo izlaze S i C_{out} čije algebarske funkcije jesu:

$$S = A \oplus B \oplus C_{in} \quad \textbf{(64)}$$

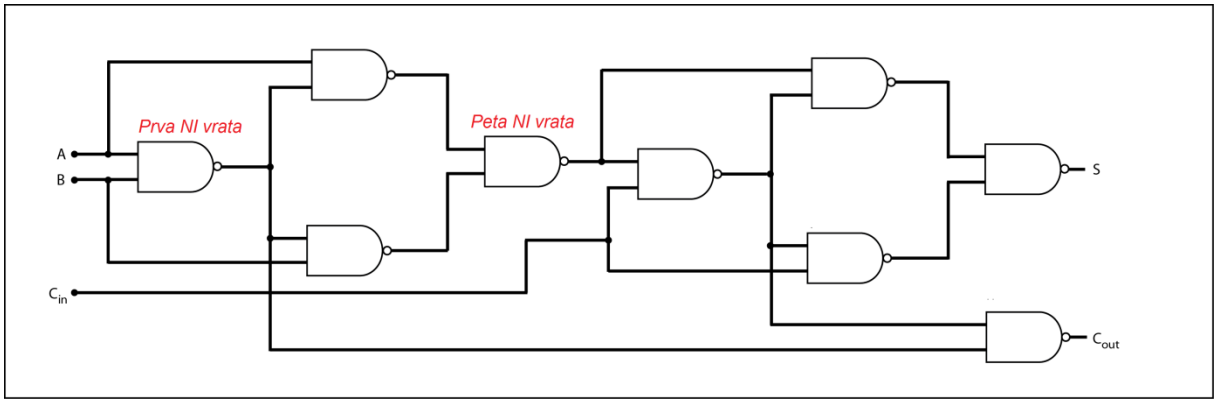
$$C_{out} = A \cdot B + Cin\ (A \oplus B). \quad \textbf{(65)}$$

Realizacija pomoću NI vrata svodi se da sklop EX-ILI zamijenimo s NI logičkim sklopom, a za to je potrebno četiri logičkih NI vrata za zamjenu jednog EX-ILI logičkog sklopa. Za zamjenu dvoja EX-ILI vrata sveukupno nam je potrebno osam NI vrata, bolje rečeno za realizaciju izlaza S. Za realizaciju izlaza C_{out} koristit ćemo drukčiji princip, tj. koristit ćemo De Morganov teorem. Tako ćemo izlazni signal prvih NI vrata i petih NI vrata dovesti na ulaz

NI vrata i time dobiti izlaz C_{out}. Promotrimo izraz za C_{out} i izvršimo duplu negaciju izraza, time nismo promijenili C_{out}, prisjetimo se Tablica 15. Nadalje ćemo disjunkciju po De Morganovom teoremu (Tablica 16) zamijeniti konjunkcijom i time dobiti sljedeći izraz:

$$\begin{aligned} C_{out} &= A \cdot B + Cin\,(A \oplus B) \\ &= \overline{\overline{A \cdot B + Cin\,(A \oplus B)}} \\ &= \overline{\overline{(A \cdot B)} \cdot \overline{Cin\,(A \oplus B)}}. \end{aligned} \quad \textbf{(66)}$$

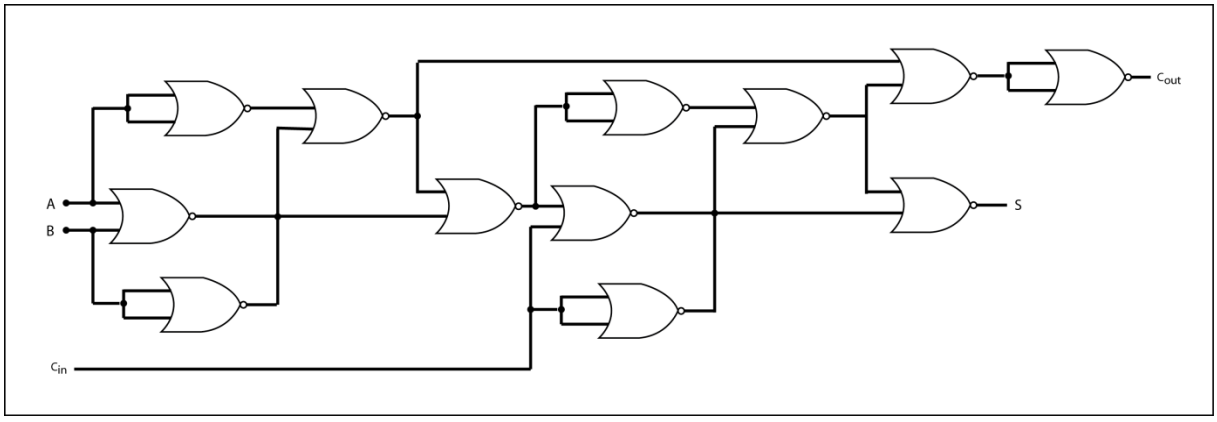
Ako promotrimo Slika 42., uvidjet ćemo da prvi član ovog izraza $\overline{(A \cdot B)}$ jest izlaz prvih NI vrata, dok drugi član izraza $\overline{Cin\,(A \oplus B)}$ izlaz je petih NI vrata. Tako je je potrebno samo ta dva izraza spojiti s jednim NI vratima.



Slika 42. Izvedba potpunog zbrajala pomoću NI vrata

7.2.2. Izvedba potpunog zbrajala pomoću NILI vrata

Isti principi izvedbe s NI vratima vrijede i za izvedbu s NILI vratima. Ova se izvedba rjeđe koristi jer je za tu izvedbu potrebno više NILI logičkih sklopova nego s NI izvedbom.



Slika 43. Izvedba potpunog zbrajala s NILI vratima

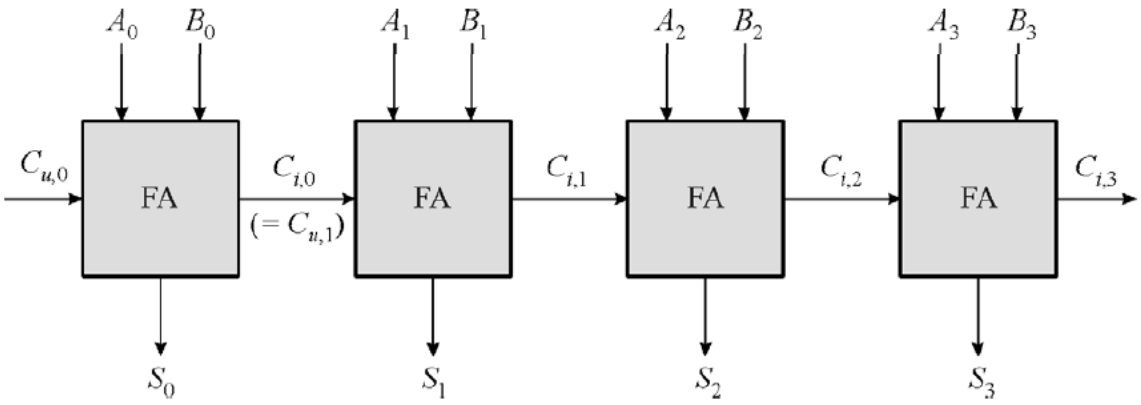
7.3. Zbrajanje višebitnih brojeva

Potpuno zbrajalo prikazano u prethodnom poglavlju u stvari je jednobitno potpuno zbrajalo. Ako želimo zbrajati dva višebitna binarna broja, onda moramo upariti više potpunih zbrajala u jedan logički sklop. Za takvo zbrajalo kažemo da zbraja dva n-bitna broja A i B, gdje je n–broj bitova najvećeg broja. Postoje razne metode spajanja jednobitnih zbrajala u višebitno zbrajalo. Naravno, ta sva zbrajala su standardizirana i integrirana u čip, a najpoznatije metode su:

- zbrajalo s postepenim prijenosom (engl. *Ripple-carry adder*)
- zbrajalo s izračunatim prijenosom (engl. *Carry-look ahead adder*).

7.3.1. Zbrajalo s postepenim prijenosom

Metoda spajanja potpunih zbrajala gdje se izlazni *carry bit* (C_{out}) spaja na *ulazni carry bit* (C_{in}) drugog zbrajala zove se *ripple – carry metoda* ili metoda s postepenim prijenosom. Kod ove konstrukcije operacija zbrajanja izvršava se paralelno, tj. ulazi binarnih brojeva istovremeno dobiju signal na ulaz potpunog zbrajala, dok se prijenos *carry* signala širi serijski, tj. unutar zbrajala bitovi binarnog broja se zbroje i „čeka“ se *carry* signal na ulazu od prethodnog zbrajala. Kada taj signal dođe, onda se može izvršiti operacija zbrajanja, a *carry* signal tog zbrajala ide na sljedeće potpuno zbrajalo koje je u nizu i koje isto tako mora „čekati“ *carry* signal tog zbrajala kako bi izvršio svoju operaciju zbrajanja. Ovakva je konstrukcija vrlo jednostavna, što omogućava brzu izradu, no sa sobom vuče nedostatke.



Slika 44. Prikaz 4-bitnog potpunog zbrajala s postepenim prijenosom

Glavni nedostatak ovakve konstrukcije jest vrijeme potrebno za obavljanje operacije, tj. ima jako veliko kašnjenje, a razlog tome je što zadnje u nizu zbrajalo mora čekati *carry* signal

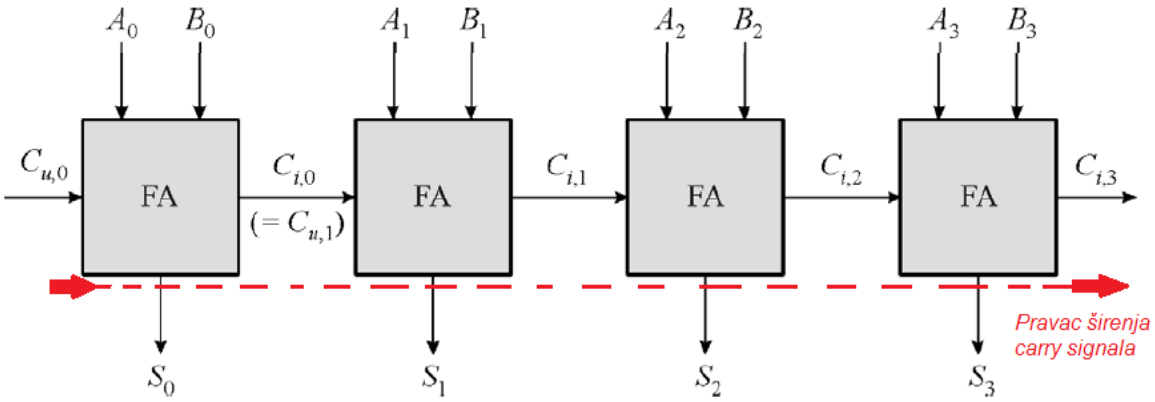
kako bi dovršilo operaciju, a taj signal ide od prvog do zadnjeg zbrajala prolazeći kroz sva zbrajala i svako zbrajalo generira kašnjenje *carry* signala. To znači da signal mora proći kroz n- EX-ILI vrata i kroz n-logičkih blokova koji generiraju *carry* signal.

Kašnjenje za najgori slučaj možemo izračunati pomoću jednadžbe:

$$t_z = (n\text{-}1) \cdot t_c + t_s$$

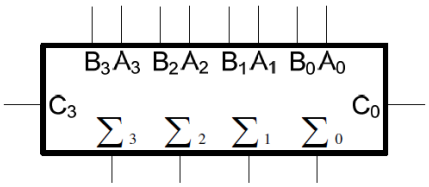
gdje su:

- t_z – vrijeme kašnjenja
- n – broj potpunih zbrajala u sklopu
- t_c – vrijeme koje je potrebno da signal prođe kroz oba EX-ILI, I i ILI vrata potpunog zbrajala
- t_s – vrijeme potrebno da signal prođe kroz oba EX-ILI vrata unutar potpunog zbrajala, tj. vrijeme potrebno da se generira izlazni signal.



Slika 45. Prikaz širenja *carry* signala

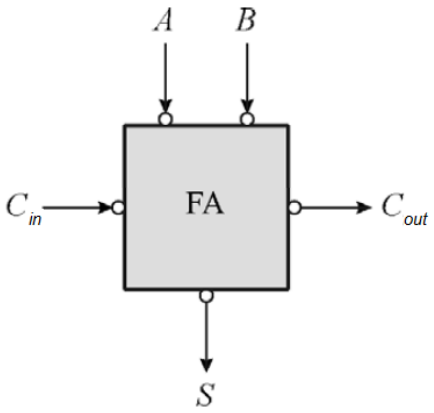
Primjer višebitnog potpunog zbrajala je čip IC 7483, obično 4-bitno zbrajalo, shematski prikazanog na slici. Ovakva konstrukcija omogućava kaskadiranje 4-bitnog zbrajala i time se omogućava zbrajanje višebitnih brojeva (N· 4), gdje je N broj čipova. Kaskadiranje se provodi na način da izlazni *carry* signal (C_4) spajamo na ulazni *carry* signal drugog čipa. Ako spojimo tako dva čipa, dobivamo 8-bitno zbrajalo, što znači da su nam za 12-bitni broj potrebna 3 čipa itd.



Slika 46. Čip 7483

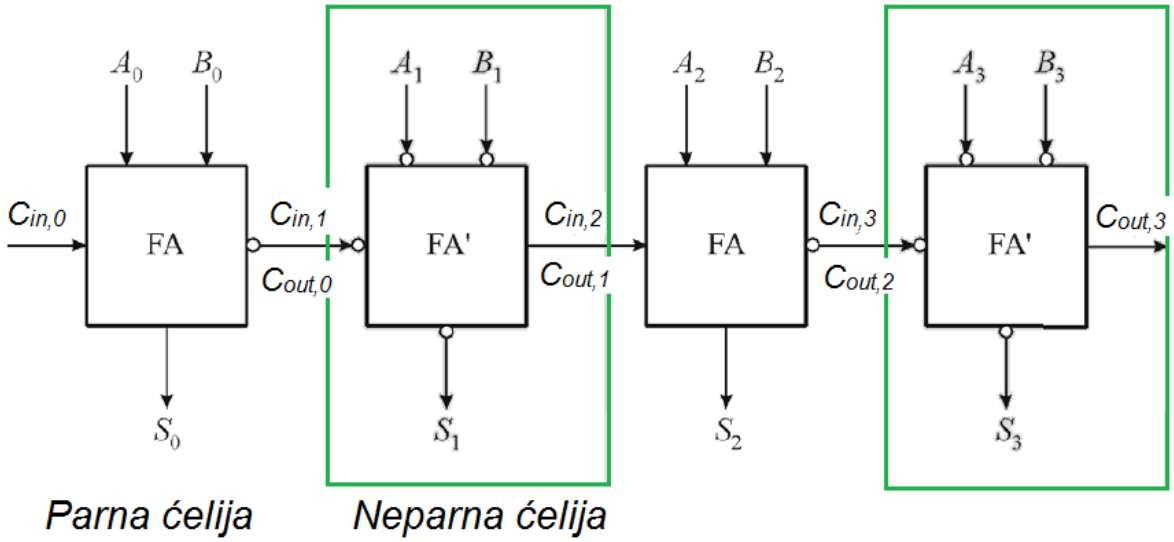
Isto tako, jedna vrsta konstrukcije višebitnog zbrajala jest da se prvo u nizu potpuno zbrajalo zamijeni polu-zbrajalom budući da nema *carry* signala, a takvo zbrajalo koristi se kada sa sigurnošću znamo da na to zbrajalo dolazi najmanje važan bit (engl. *Least significant bit*). Takvo zbrajalo gubi mogućnost kaskadiranja, tj. samo na njega se može kaskadirati zbrajalo tipa IC 7483.

Kako je potreba za brzinom digitalnih sklopova rasla, tako se je razmišljalo kako ubrzati višebitna zbrajala. Jedan od načina jest primjena zbrajala s inventiranim ulazima i izlazima prikazanog na Slika 47.



Slika 47. Potpuno zbrajalo s invertiranim ulazima i izlazima

Brzina višebitnog zbrajala može se povećati ako se koriste obična zbrajala sa zbrajalima s inverterima, tj. ako se koristi svojstvo inverzije i time se smanjuje broj logičkih vrata i ubrzava širenje *carry* signala. Takva se zbrajala spajaju kao na Slika 48.



Slika 48. Potpuno zbrajalo s kombinacijom invertiranih ulaza/izlaza

7.3.2. Zbrajalo s izračunatim prijenosom

Razvojem digitalnih sklopova rasla je potreba za bržim digitalnim sklopovima. Osmišljavali su se razni načini spajanja kako bi se postigle što veće brzine rada višebitnog potpunog zbrajala. Najslabija karika, bolje rečeno izazov je ubrzati carry signal, tj. preljev. Radi povećanja brzine carry signala osmišljeno je zbrajalo s izračunatim prijenosom (engl. *carry-look ahead adder*). Takva je metoda idealno jednostavna, ali sklopovski komplicirana. Zbrajalo s izračunatim prijenosom pokušava „predvidjeti" iznos *carry* signala, tj. nizom jednostavnih algoritama pokušava izračunati *carry* signal. Vratimo se natrag na potpuno zbrajalo i njegovu tablicu stanja, samo što će u ovom primjeru biti prikazan samo *carry* izlaz.

Tablica 38. Tablica stnja oba *carry* signala

A_i	B_i	C_i	C_{i+1}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Booleov izraz za C_{i+1} je:

$$C_{i+1} = A \cdot B + C_i. \quad \textbf{(67)}$$

Ako pogledamo Tablica 38., primijetimo zadnja dva retka i uvidjet ćemo da će se ulazni signal mijenjati, ali izlazni carry signal ostaje u jedinici, tj. ne mijenja se. Uvidimo da to vrijedi samo kada su ulazne varijable A_i i B_i u logičkoj jedinici.

Iz toga slijedi izraz:

$$G_i = A_i \cdot B_i. \quad \textbf{(68)}$$

Ovaj izraz, tj. G_i predstavlja generirajući član (engl. *generate bit*). Iz glavne jednadžbe vidimo da će, ako je izraz G_i u logičkoj jedinici, C_{i+1} isto biti u logičkoj jedinici (teorem o anihilaciji, Tablica 14.). Iz tog slijedi zaključak da se sigurno zna da će *carry* signal biti u jedinici ako su obje binarne znamenke višebitnog broja u jedinici. Zato se G_i zove generirajući član jer generira *carry* signal i šalje ga sljedećem potpunom zbrajalu u nizu bez čekanja *carry* signala prethodnog zbrajala. Time je omogućeno da sljedeće potpuno zbrajalo može dovršiti svoj izračun jer je neovisan o nadolazećem *carry* signalu C_i, čime je ubrzan rad višebitnog zbrajala. Nadalje, ako opet promatramo Tablica 38., tj. od 3. do 6. retka, pogledamo

jednadžbu za $C_{i+1,}$ uvidjet ćemo da izlazni carry signal C_{i+1} ovisi o ulaznom *carry* signalu $C_{i,}$ zapisan Booleovom jednadžbom:

$$P_i = (A \oplus B) \cdot C_{i.} \quad \textbf{(69)}$$

Ovaj izraz $P_{i,}$ predstavlja propagirajući član (engl. *propagate bit*). Ako pogledamo tablicu, do prijenosa *carry bita* C_i dolazi ako je A_i ili B_i u jedinici, tj. ako je jedan od ulaza u logičkoj jedinici. P_i se zove propagirajući član jer „propagira" prijenos s prethodnog *carry* signala, tj. ako je bilo koji od ulaza u logičkoj jedinici, čeka se na *carry* bit i onda se taj *carry bit* dalje prosljeđuje, zbog tog svojstva da propagira ako je A ili B u jedinici mogu se EX-ILI vrata zamjeniti ILI vratima..

$$P_i = (A + B) \cdot C_i \quad \textbf{(70)}$$

Razlog je što su ILI vrata brži logički sklop od EX-ILI vrata, ako su oba ulaza u jedinici neće se „pokvariti" izraz C_{i+1} jer će se u tom slučaju dogoditi anihilacija G_i člana naprema P_i članu. Često se koriste EX-ILI vrata za višebitna zbrajala (n > 4).

Uvrštavanjem G_i i P_i u jednadžbu (67) dobijemo:

$$C_{i+1} = G_i + P_i \cdot C_{i.} \quad \textbf{(71)}$$

Zamislimo 4-bitno zbrajalo prikazano shemom na Slika 49. Svaki *carry* signal ima svoju Booleovu jednadžbu prikazanu pomoću P_i i G_i članova.

$$C_1 = G_0 + P_0 \cdot C_0 \quad \textbf{(72)}$$

$$C_2 = G_1 + P_1 \cdot C_1 \quad \textbf{(73)}$$

$$C_3 = G_2 + P_2 \cdot C_2 \quad \textbf{(74)}$$

$$C_4 = G_3 + P_3 \cdot C_3 \quad \textbf{(75)}$$

Ukoliko uvrstimo C_1 u C_2, pa C_2 u C_3 i C_3 u $C_{4,}$ dobijemo sljedeće relacije:

$$C_1 = G_0 + P_0 \cdot C_0 \quad \textbf{(76)}$$

$$C_2 = G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0) \quad \textbf{(77)}$$

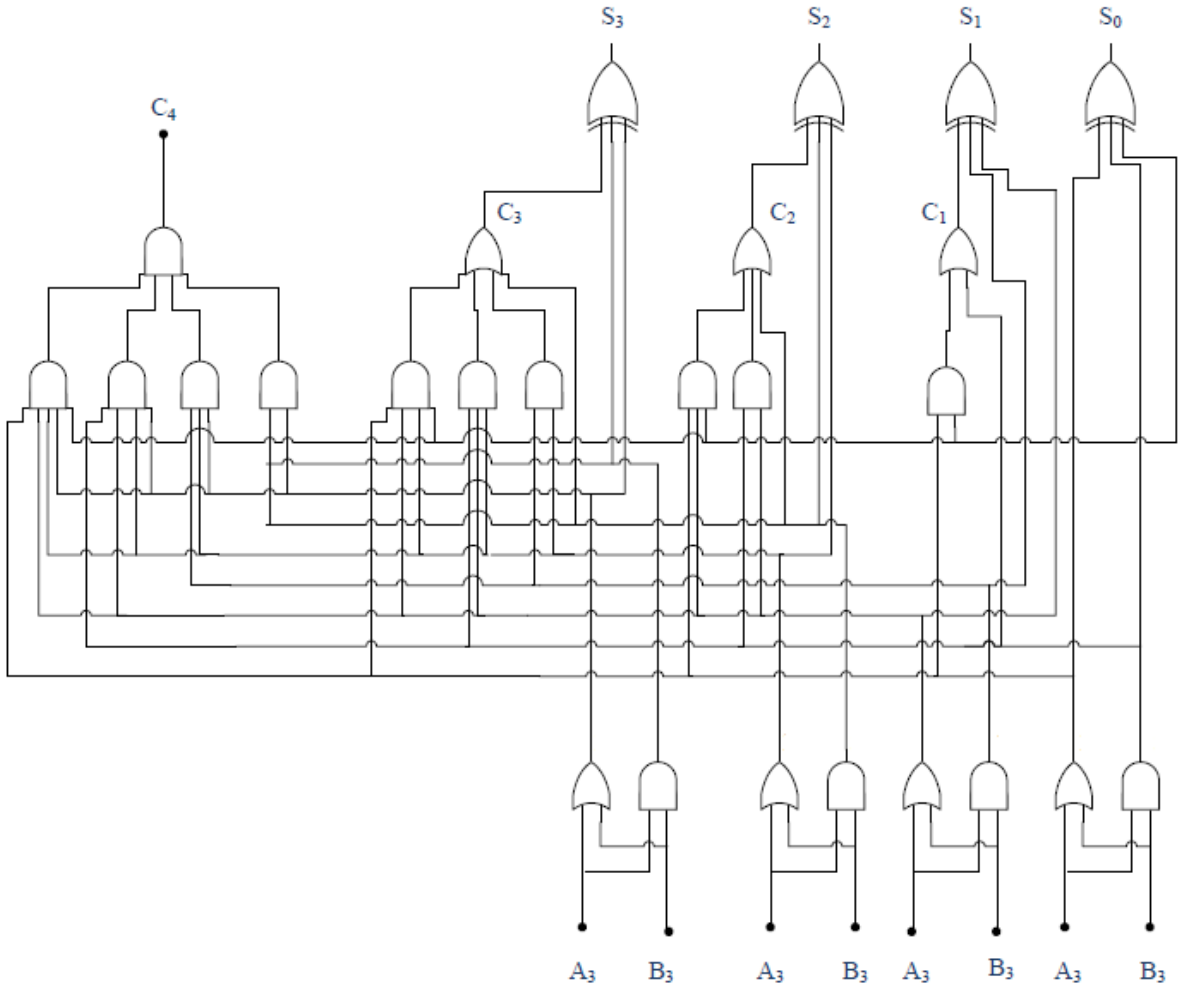
$$C_3 = G_2 + P_2 \cdot (G_1 + G_0 \cdot P_1 + P_0 \cdot P_1 \cdot C_0) \quad \textbf{(78)}$$

$$C_4 = G_3 + P_3 \cdot (G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + P_0 \cdot P_1 \cdot P_2 \cdot C_0 . G_0 . P_1 . P_2 + P_0 . P_1 . P_2 . C_0)$$

$$C_4 = G_3 + G_2 . P_3 + G_1 . P_2 \cdot P_3 + G_0 . P_1 . P_2 . P_3 + P_0 . P_1 . P_2 . P_3 . C_{0.} \quad \textbf{(79)}$$

Iz jednadžbi možemo vidjeti da *carry* signal, koji je potreban da bi se izračunala funkcija, početni je *carry* signal C_0. Znajući da znamenke binarnih brojeva sve dolaze paralelno u zbrajalo, može se brzo dobiti vrijednost G_i i P_i te iz toga slijedi da se prijenos može brzo izračunati bez potrebe da se čeka prethodni *carry* signal. Naravno, to za sobom potеže

kompliciraniji logički sklop i drukčije ožičenje, ali zato ima za prednost brže računanje, tj. zbrajanje.



Slika 49. Prikaz sheme *carry-look ahead* zbrajala

8. ZAKLJUČAK

Do sada su opisani fundamentalna znanja za projektiranje potpunog zbrajala. Kaskadnim povezivanjem više potpunih zbrajala dobiva se višebitno potpuno zbrajalo. Na *protoboardu* (eksperimentalna ploča SD-120) će biti prikazana fizička implementacija 4-bitnog potpunog zbrajala. Zbrajalo će biti konstruirano isključivo od NI vrata, korištenjem čipa 74HC00, time se želi pokazati primjer industrijske proizvodnje logičkih sklopova. Pomoću LED dioda, spojenih u seriju s otpornikom 330 Ω, biti će prikazani ulazni i izlazni binarni brojevi, tj. rezultat zbrajanja. Cilj zaključka je usporediti prednosti i mane konstruiranja logičkih sklopova isključivo s NI vratima naprema konstruiranje kombinacijom ostalih logičkih sklopova, u ovom slučaju konstruiranje s EX-ILI, I, ILI vratima.

Usporedbom cijena dobijemo da je NI varijanta skuplja za 4,17 kuna (kn), tj. za NI varijantu je potrebno izdvojiti 11,92 kn, a za običnu 7,75 kn. Kod NI varijante su iskorišteni svi čipovi kao i sva NI vrata dok kod obične varijante ostalo je neiskorišteno po jedna logička vrata, tj. jedna EX-ILI, I, ILI vrata su ostala neiskorištena. Ako gledamo s financijske strane ostalo je 1,58 kn neiskorišteno, dok je kod NI izvedbe sav financijski potencijal iskorišten.

Prednost korištenja isključivo NI vrata:

- Lakše pozicioniranje i ožičenje čipova
- Isto vrijeme prolaska signala kroz sva vrata
- Potpuno iskorištenje čipa
- Lakše kombiniranje ulaza i izlaza čipa

Mane korištenja isključivo NI vrata:

- Skuplja varijanta
- Potrebno je više logičkih vrata
- Potrebno mu je više vremena za davanje rezultata tj. signal mora proći više logičkih vrata
- Zauzima više mjesta zbog većeg broja čipova

Tu vidimo da ima više parametara za koje bi se konstruktor logičkih sklopova trebao odlučiti, vidi se da NI varijanta zauzima više mjesta, ali je lakša za projektiranje, tu vidimo dilemu prostora tj. veličine potrebne pločice. NI varijanta je sporija dok u slučaju kvara čipa lako se

zamjenjuje i lako se pronalazi kvar, tu dolazi aspekt održavanja i brzine. Kod financijske strane NI varijanta je skuplja, ali lakše je kupiti maleni višak jedne vrste čipova, nego kada se ima višak više čipova različite vrste, tu dolazi do izražaja financijska strana projektiranja. Tako ostaje na konstruktoru logičkih sklopova da sam izabere koji parametri su mu bitni, a koji sekundarni.

PRILOZI

I. CD-R disc

II. Dokumentacija korištenih elektroničkih komponenata

LITERATURA

[1] Kraut, B.: Strojarski priručnik, Tehnička knjiga Zagreb, 1970.

[2] Decker, K. H.: Elementi strojeva, Tehnička knjiga Zagreb, 1975.

[3] Herold, Z.: Računalna i inženjerska grafika, Zagreb, 2003.

[4] http://www.computerhistory.org/timeline/computers/

[5] http://www.explainthatstuff.com/historyofcomputers.html

[6] WIKIPEDIA: Logički sklopovi, engl. *Logic gate*
https://hr.wikipedia.org/wiki/Logički_sklopovi,
https://en.wikipedia.org/wiki/Logic_gate

[7] WIKIPEDIA: Arapski brojevi, https://bs.wikipedia.org/wiki/Arapski_brojevi

[8] WIKIPEDIA: Gottfried W. Leibniz,
https://en.wikipedia.org/wiki/Gottfried_Wilhelm_Leibniz

[9] WIKIPEDIA: George Boole, https://en.wikipedia.org/wiki/George_Boole

[10] Studenska stranica PMF-a, povijest računala,
http://web.studenti.math.pmf.unizg.hr/~bozana/povijest.html

[11] Javno dostupna pdf predavanja kolegija Digitalna logika, FER,
http://www.fer.unizg.hr/_download/repository/DL07_08-09_Aritm2._skl.pdf

[12] ELECTRONICS HUB: Half-adder, Full-adder, ripple carry adder
http://www.electronicshub.org/
http://www.electronicshub.org/half-adder-and-full-adder-circuits/

[13] CIRCUITS TODAY: Boolean logic, Logic gates, Full & Half adders,
http://www.circuitstoday.com/logic-gates
http://www.circuitstoday.com/half-adder-and-full-adder
http://www.circuitstoday.com/half-adder-and-full-adder

[14] Bogdanivić, D. : Aritmetičko logička jedinica (ALU), završni rad, Osijek, rujan 2010.

[15] Čupić, Marko: Digitalna elektronika i digitalna logika, zbirka riješenih zadataka, Kingen, Zagreb 2006.

[16] Paunović, Stanko: Digitalna elektronika, Školska knjiga, Zagreb, 1999.

[17] Youtube kanal Neso Academy, https://www.youtube.com/watch?v=aLUY-s7LSns

[18] Internet

Logička shema 4-bitnog zbrajala